2756

RS

#4

520.37728X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): INOHARA, et al.

Serial No.: 09/416,961

Filed: October 13, 1999

Title: METHOD AND APPARATUS FOR IMPLEMENTING EXTENSIBLE NETWORK-ATTACHED SECONDARY STORAGE

Group:

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

November 23, 1999

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on:

Japanese Patent Application No. 10-293277
Filed: October 15, 1998

A certified copy of said Japanese Patent Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/ssr
Attachment

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されてい る事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: １９９８年１０月１５日

出 願 番 号
Application Number: 平成１０年特許願第２９３２７７号

出 願 人
Applicant (s): 株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

１９９９年１０月 ８日

特許庁長官
Commissioner,
Patent Office 近 藤 隆 彦

出証番号 出証特平１１－３０６８０５

【書類名】　特許願

【整理番号】　H98023031A

【提出日】　平成10年10月15日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F　3/06

【発明の名称】　拡張型ネットワーク接続二次記憶方法及び装置

【請求項の数】　35

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

【氏名】　猪原　茂和

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

【氏名】　西澤　格

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

【氏名】　渡邉　直企

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

【氏名】　富田　亜紀

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

【氏名】　マシエル　フレデリコ

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地

株式会社日立製作所中央研究所内

【氏名】　小田原　宏明

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地

株式会社日立製作所中央研究所内

【氏名】　佐川　暢俊

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地

株式会社日立製作所中央研究所内

【氏名】　杉江　衛

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100068504

【弁理士】

【氏名又は名称】　小川　勝男

【電話番号】　03-3212-1111

【手数料の表示】

【予納台帳番号】　013088

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　拡張型ネットワーク接続二次記憶方法及び装置

【特許請求の範囲】

【請求項１】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、

該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、

該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第１のコンピュータに提供し、

第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を受けた後、第１のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求を受け、該オブジェクトアクセスモジュールを実行することにより該要求によるオブジェクト単位入出力を行う

ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項２】

前記オブジェクトアクセスモジュールが、オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・タグ（取り出すべきデータの種類の指定）、のいずれかの指定から、対応するデータの値または対応するデータのブロック内での位置を得る請求項１記載の拡張型ネットワーク接続二次記憶方法。

【請求項３】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、

該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、

該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第１のコンピュータに提供するコンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータとして動作し、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を、該二次記憶装置に登録または削除する

拡張型ネットワーク接続二次記憶方法。

【請求項４】

前記オブジェクトアクセスモジュールが、オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・タグ（取り出すべきデータの種類の指定）、のいずれかの指定から、対応するデータの値または対応するデータのブロック内での位置を得る請求項３記載の第１のコンピュータ。

【請求項５】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶装置が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第１のコンピュータに提供し、
該機能を、第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから、該二次記憶装置に送って実行することにより
、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュール。

【請求項６】

オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・タグ（取り出すべきデータの種類の指定）、のいずれかの指定から、
対応するデータの値または対応するデータのブロック内での位置を得る請求項５記載のオブジェクトアクセスモジュール。

【請求項７】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶装置が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第１のコンピュータに提供し、
該オブジェクトが該二次記憶に格納されている方法を示すデータ（オブジェクト記述データ）を第１のコンピュータまたは第１のコンピュータと異なる第２のコ

ンピュータから受けた後、第１のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求を受け、
該オブジェクト記述データを用いて該オブジェクトの該二次記憶上の位置を特定することにより該要求の入出力を行う、
ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項８】

前記オブジェクト記述データが、オフセットとサイズによって、ブロック内データの属性またはブロック間参照関係を指定するデータである請求項７記載の拡張型ネットワーク接続二次記憶方法。

【請求項９】

前記オブジェクト記述データが、構文解析プログラム（パーザ）またはパーザ生成用文法によって、ブロック内データの属性またはブロック間参照関係を指定するデータである請求項７記載の拡張型ネットワーク接続二次記憶方法。

【請求項１０】

前記オブジェクト記述データが、１つ以上のブロックの特定部分に格納されているデータが、特定の値またはパターンであるかによって、前記オブジェクトのファイルフォーマットを特定するデータである
請求項７記載の拡張型ネットワーク接続二次記憶方法。

【請求項１１】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能

とを第１のコンピュータに提供するコンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータの機能を有し、該オブジェクトが該二次記憶に格納されている方法を示すデータ（オブジェクト記述データ）を該二次記憶装置に登録または削除する
ことを特徴とするコンピュータ。

【請求項１２】

前記オブジェクト記述データが、オフセットとサイズによって、ブロック内のデータの並びまたはブロック間の参照関係を指定するデータである請求項１１記載のコンピュータ。

【請求項１３】

前記オブジェクト記述データが、構文解析プログラム（パーザ）またはパーザ生成用文法によって、ブロック内のデータの並びまたはブロック間の参照関係を指定するデータである請求項１１に記載のコンピュータ。

【請求項１４】

前記オブジェクト記述データが、１つ以上のブロックの特定の部分に格納されているデータが、特定の値またはパターンであるかによって、前記オブジェクトのファイルフォーマットを特定するデータである請求項１１に記載のコンピュータ。

【請求項１５】

前記オブジェクトアクセスモジュールが、請求項７乃至１４記載のオブジェクト記述データからオブジェクトが二次記憶に格納されている方法を得る拡張型ネットワーク接続二次記憶方法。

【請求項１６】

前記オブジェクトアクセスモジュールが、請求項７乃至請求項１４記載のオブジェクト記述データからオブジェクトが二次記憶に格納されている方法を得る請求項３記載の第１のコンピュータ。

【請求項１７】

請求項７乃至請求項１４に記載のオブジェクト記述データからオブジェクトが二次記憶に格納されている方法を得る請求項５記載のオブジェクトアクセスモジ

ュール。

【請求項１８】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、

該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、

該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、コンピュータシステムの二次記憶装置であって、

ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、

該オブジェクトアクセスモジュールを用いて該高機能入出力を実現するモジュール（機能モジュール）を、第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから受けた後に、第１のコンピュータから該高機能入出力の要求を受け、

該機能モジュールを実行することにより該要求の入出力を行う

ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項１９】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第１のコンピュータに提供するコンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータとして動作し、
該二次記憶装置が、オブジェクト単位入出力の機能をブロック単位入出力の機能を用いて実現するモジュール（オブジェクトアクセスモジュール）を保持し、
該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を、該コンピュータが登録または削除する
ことを特徴とするコンピュータ。

【請求項２０】
１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、
第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから、該二次記憶装置に送られ、
該二次記憶装置で実行され、
ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するモジュール（オブジェクトアクセスモジュール）を利用して該高機能入出力を実現する

ことを特徴とするプログラム・モジュール。

【請求項２１】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、

該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、

該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、

該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するモジュール（オブジェクトアクセスモジュール）を保持し、

該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、

該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定する、

ことを特徴とする二次記憶装置及び該二次記憶装置上のプロテクションモジュール。

【請求項２２】

前記プロテクションモジュールが、前記コンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータによって、前記二次記憶装置へ登録、または前記二次記憶装置から削除される請求項２１記載のプロテクションモジュール。

【請求項２３】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備

え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、
該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュール）を保持し、
該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、
該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定するモジュール（プロテクションモジュール）を、第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから受ける、
ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項２４】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出

力（高機能入出力）の機能と、を第１のコンピュータに提供するコンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータとして動作し、
該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、
該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、
該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定するプロテクションモジュールを該二次記憶装置に登録または削除する
ことを特徴とするコンピュータ。

【請求項２５】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、
該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、
該オブジェクトアクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供する、
ことを特徴とする二次記憶装置上のロックモジュール。

【請求項２６】

前記ロックモジュールは、前記第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータによって、前記二次記憶装置へ登録、または前記二次記憶装置から削除される請求項２５記載のロックモジュール。

【請求項２７】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、

該二次記憶が、複数の格納単位（ブロック）からなり、

該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、

該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第１のコンピュータに提供するコンピュータシステムの二次記憶装置であって、

該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、

該オブジェクトアクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供するロックモジュールを、第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから受ける、

ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項２８】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、

第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第１のコンピュータに提供するコンピュータシステム内で動作する第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータとして動作し、
該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、
該オブジェクトアクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供するロックモジュールを、該二次記憶装置に登録または二次記憶装置から削除する、
ことを特徴とするコンピュータ。

【請求項２９】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、１つの第２のコンピュータ（管理マシン）と、該第１のコンピュータと該二次記憶装置と該管理マシンを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
第２のコンピュータが該二次記憶装置のリストを保持しており、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第１のコンピュータに提供し、
該高機能入出力を実現するプログラム・モジュール（モジュール）を第１のコン

ピュータが第２のコンピュータに送り、
第２のコンピュータが、該モジュールを受け取り、該リストに保持された該二次記憶装置の一部または全部に該モジュールを送り、
該二次記憶装置が該モジュールを受け取り、
第１のコンピュータが該高機能入出力の要求を二次記憶装置に送り、
該二次記憶装置が該モジュールを起動することにより該高機能入出力を行う、
ことを特徴とする管理マシン。

【請求項３０】

前記プログラム・モジュールを前記二次記憶装置向けにコンパイルするコンパイラを備え、
第１のコンピュータから受け取った前記モジュールを、該コンパイラによってコンパイルし、
該二次記憶装置の一部または全部にコンパイル済みモジュールを送る請求項２９記載の管理マシン。

【請求項３１】

前記二次記憶装置の機種データを保持し、
前記二次記憶装置の機種毎に、モジュールをコンパイルする１つ以上のコンパイラを備え、
第１のコンピュータから受け取った前記モジュールを、該１つ以上のコンパイラによって送付先の二次記憶装置向けにコンパイルし、
該二次記憶装置の一部または全部にコンパイル済みモジュールを送る請求項３０記載の管理マシン。

【請求項３２】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、１つの第２のコンピュータ（管理マシン）と、該第１のコンピュータと該二次記憶装置と該管理マシンを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
第２のコンピュータが該二次記憶装置のリストを保持しており、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記

憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第１のコンピュータに提供し、
前記高機能入出力を実現するプログラム・モジュール（モジュール）を第１のコンピュータが第２のコンピュータに送り、
第２のコンピュータが、該モジュールを受け取り、該リストに保持された前記二次記憶装置の一部または全部に該モジュールを送り、
該二次記憶装置が該モジュールを受け取り、
第１のコンピュータが該高機能入出力の要求を二次記憶装置に送り、
該二次記憶装置が該モジュールを実行することにより該高機能入出力を行う
ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項３３】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第１のコンピュータに提供し、
第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）または該高機能入

出力を実現するプログラム・モジュールを受け、
該二次記憶装置が、該モジュールを高速実行が可能な実行プログラムへコンパイルするコンパイラを備え、
該二次記憶装置が該コンパイラで該モジュールをコンパイルし、
第１のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求または高機能入出力の要求を受け、
該コンパイルしたモジュールを実行することにより該要求の入出力を行う、
ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項３４】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、
該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第１のコンピュータに提供し、該高機能入出力の機能として、第１のコンピュータの要求に応じて
１つのオブジェクトの複数の部分と、該部分を格納する場所である二次記憶装置との対応関係（ストライプ構成情報）を返す機能を備える、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項３５】

１つ以上の第１のコンピュータと、１つ以上の二次記憶装置と、該第１のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、
第１のコンピュータが１つ以上の応用プログラムを動作させ、

該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、
該二次記憶が、複数の格納単位（ブロック）からなり、
該二次記憶が、該応用プログラムが用いる１つ以上の応用向けデータ（オブジェクト）を１つ以上のブロックに格納しており、
ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第１のコンピュータに提供する手段と、
第１のコンピュータまたは第１のコンピュータと異なる第２のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を受ける手段と、
該オブジェクトアクセスモジュールを受けた後、第１のコンピュータから該オブジェクトアクセスモジュールに対するオブジェクト単位入出力の要求を受ける手段と、
該オブジェクトアクセスモジュールを実行する手段と、
を有することを特徴とする拡張型ネットワーク接続二次記憶装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はコンピュータシステムに関し、特に応用プログラムに適した新たな機能を追加・拡張可能な二次記憶方法及び装置に関する。

【０００２】

【従来の技術】

今日のコンピュータシステムの主要な構成要素は、コンピュータ（プロセッサとメモリ、周辺機器からなる）とネットワークと二次記憶装置（ストレージ）である。これまでストレージは、コンピュータに付属する装置として存在することが多かったが、近年、このような状況が変化しつつある。

【０００３】

第１に、ネットワークの普及により、複数のコンピュータがストレージを共有する機会が増えた。他のコンピュータからのネットワーク経由のストレージ入出力

が、ストレージを接続したコンピュータのプロセッサ処理能力がボトルネックとなって滞る事態を招いている。第２に、ストレージ容量およびストレージに対する要求スループットは、年々増大している。「データウェアハウス用途のストレージ容量の要求は、９ヶ月で２倍になる」という予測（Ｇｒｅｇ’ｓ　Ｌａｗ）もある。このため、１つのコンピュータに接続するストレージ数が増大し、やはりコンピュータがストレージ入出力のボトルネックとなる恐れが出てきている。第３に、ハードディスク制御用ＬＳＩの高集積化の進行によって、ストレージの高機能化の可能性が増大している。

【０００４】

これらの背景から、ストレージの制御用ＬＳＩに新たな機能を付加することが考えられている。新たな機能の候補は、ネットワークインタフェースと、応用プログラム向けの高度機能である。

【０００５】

ネットワークインタフェースをストレージが備えることにより、ストレージをネットワークに直結することができる。これによりストレージは、複数のコンピュータからの入出力要求を、１つのコンピュータを介することなく、直接受けることができる。

【０００６】

現在、ストレージとコンピュータとの間の最も代表的なインタフェースはブロック入出力であるが、これよりも応用毎の高度機能（例えば、ソーティング、画像処理、データベースシステムにおける基本演算（例えば、選択処理、写像処理、結合処理、集計処理等）をストレージが備えることにより、コンピュータのプロセッサ処理の一部をストレージが受け持つことが可能となる。

【０００７】

ネットワークインタフェースを備え、かつファイルシステムの一部分の機能を備えたストレージの提案の代表例としては、Ｇａｒｔｈ　Ａ.Ｇｉｂｓｏｎ他著の文献”Ａ　Ｃｏｓｔ－Ｅｆｆｅｃｔｉｖｅ、Ｈｉｇｈ－Ｂａｎｄｗｉｄｔｈ　Ｓｔｏｒａｇｅ　Ａｒｃｈｉｔｅｃｔｕｒｅ”（Ｐｒｏｃｅｅｄｉｎｇｓ　ｏｆ　ｔｈｅ　８ｔｈ　Ｃｏｎｆｅｒｅｎｃｅ　ｏｎ　Ａｒｃｈｉｔｅｃｔｕｒａｌ

Ｓｕｐｐｏｒｔ ｆｏｒ Ｐｒｏｇｒａｍｍｉｎｇ Ｌａｎｇｕａｇｅｓ ａｎｄ Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍｓ、１９９８年ＡＣＭ発行；以後文献１と記す）とＳｔｅｖｅｎ Ｒ.Ｓｏｌｔｉｓ他著の文献”Ｔｈｅ Ｇｌｏｂａｌ Ｆｉｌｅ Ｓｙｓｔｅｍ”（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ Ｆｉｆｔｈ ＮＡＳＡ Ｇｏｄｄａｒｄ Ｓｐａｃｅ Ｆｌｉｇｈｔ Ｃｅｎｔｅｒ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｍａｓｓ Ｓｔｏｒａｇｅ Ｓｙｓｔｅｍｓ ａｎｄ Ｔｅｃｈｎｏｌｏｇｉｅｓ、１９９６年ＮＡＳＡ発行；以後文献２と記す）に記載のシステムがある。

【０００８】

また複数の応用を想定して、ストレージを高機能化する提案の代表例としては、Ｅｒｉｋ Ｒｉｅｄｅｌ他著の文献”Ａｃｔｉｖｅ Ｄｉｓｋｓ：Ｒｅｍｏｔｅ Ｅｘｅｃｕｔｉｏｎ ｆｏｒ Ｎｅｔｗｏｒｋ－Ａｔｔａｃｈｅｄ Ｓｔｏｒａｇｅ”（Ｔｅｃｈｎｉｃａｌ Ｒｅｐｏｒｔ ＣＭＵ－ＣＳ－９７－１９８、１９９７年Ｃａｒｎｅｇｉｅ Ｍｅｌｌｏｎ Ｕｎｉｖｅｒｓｉｔｙ発行；以後文献３と記す）、Ａｎｕｒａｇ Ａｃｈａｒｙａ他著の文献”Ａｃｔｉｖｅ Ｄｉｓｋｓ：Ｐｒｏｇｒａｍｍｉｎｇ Ｍｏｄｅｌ、Ａｌｇｏｒｉｔｈｍｓ ａｎｄ Ｅｖａｌｕａｔｉｏｎ”（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ８ｔｈ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ａｒｃｈｉｔｅｃｔｕｒａｌ Ｓｕｐｐｏｒｔ ｆｏｒ Ｐｒｏｇｒａｍｍｉｎｇ Ｌａｎｇｕａｇｅｓ ａｎｄ Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍｓ、１９９８年ＡＣＭ発行；以後文献４と記す）、Ｋｉｍｂｅｒｌｙ Ｋｅｅｔｏｎ他著の文献”Ａ Ｃａｓｅ ｆｏｒ Ｉｎｔｅｌｌｉｇｅｎｔ Ｄｉｓｋｓ（ＩＤＩＳＫｓ）”（ＳＩＧＭＯＤ Ｒｅｃｏｒｄ、Ｖｏｌｕｍｅ ２７、Ｎｕｍｂｅｒ ３、１９９８年ＡＣＭ発行；以後文献５と記す）に記載のシステムがある。

【０００９】

文献３、文献４、文献５はいずれも、ネットワーク経由で高機能を実現するプログラム・モジュールをコンピュータからストレージに転送（ダウンロード）することについて言及している。このようなモジュールを記述する言語の案としては、Ｊ.Ｇｏｓｌｉｎｇ他著の文献”Ｔｈｅ Ｊａｖａ Ｌａｎｇｕａｇｅ Ｓｐ

ｅｃｉｆｉｃａｔｉｏｎ”（１９９６年Ａｄｄｉｓｏｎ－Ｗｅｓｌｅｙ発行；以後文献６と記す）に記載の言語がある。

【００１０】

図２を用いて、従来のストレージの代表であるサーバ接続型ストレージ（Ｓｅｒｖｅｒ－Ａｔｔａｃｈｅｄ　Ｄｉｓｋ；ＳＡＤ）の構成を説明する。

【００１１】

１つ以上のＳＡＤ型ストレージ２０３は、入出力用信号線２０２によって通常１つのコンピュータ２０１と接続される。場合によっては複数のコンピュータと接続されることもある。ＳＡＤ型ストレージ２０３はストレージコントローラ２０４とディスク２０９からなり、ストレージコントローラ２０４はさらにインタフェース制御部２０５、バッファ管理部２０６、バッファメモリ２０７、ディスク制御部２０８からなる。

【００１２】

ディスク２０９は電源断後もデータを保持することが可能な記憶媒体（二次記憶）である。インタフェース制御部２０５は、外部から入出力用信号線２０２経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信を入出力用信号線２０２へ送り出す。バッファ管理部２０６は、バッファメモリ２０７の制御を行う部分である。バッファメモリ２０７は、ディスク２０９から得たデータを一時的にためておくメモリである。ディスク制御部２０８は、ディスク２０９にブロックの読み出し・書き込みを行わせる制御を行う。

【００１３】

ＳＡＤ型ストレージ２０３とコンピュータ２０１の間のインタフェースであるＳＡＤインタフェース２１０は、ブロック単位入出力を提供する。

【００１４】

図３を用いて、近年登場しつつあるネットワーク接続型ストレージ（Ｎｅｔｗｏｒｋ－Ａｔｔａｃｈｅｄ　Ｓｔｏｒａｇｅ　ＮＡＳ）の構成を説明する。

【００１５】

１つ以上のＮＡＳ型ストレージ３０３は、ネットワーク３０２によって１つ以上のコンピュータ３０１、３０１’、…と接続される。ＮＡＳ型ストレージ３０３

はストレージコントローラ３０４とディスク３０９からなり、ストレージコントローラ３０４はさらにネットワーク制御部３０５、バッファ管理部３０６、バッファメモリ３０７、ディスク制御部３０８からなる。

【００１６】

ネットワーク制御部３０５は、外部からネットワーク３０２経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信をネットワーク３０２へ送り出す。ディスク３０９、バッファ管理部３０６、バッファメモリ３０７、ディスク制御部３０８は、それぞれディスク２０９、バッファ管理部２０６、バッファメモリ２０７、ディスク制御部２０８と同等の機能を持つ。

【００１７】

ＮＡＳ型ストレージ３０３とコンピュータ３０１、３０１’、…の間のインタフェースであるＮＡＳインタフェース３１０は、ブロック単位入出力を提供する。

【００１８】

図４を用いて、従来のＳＡＤ型ストレージを拡張した高機能ＳＡＤ型ストレージの構成を説明する。

【００１９】

１つ以上の高機能ＳＡＤ型ストレージ４０３は、入出力用信号線４０２によって通常１つのコンピュータ４０１と接続される。場合によっては複数のコンピュータと接続されることもある。高機能ＳＡＤ型ストレージ４０３はストレージコントローラ４０４とディスク４０９からなり、ストレージコントローラ４０４はさらにインタフェース制御部４０５、バッファ管理部４０６、バッファメモリ４０７、ディスク制御部４０８、及び、高機能処理部４１１からなる。

【００２０】

入出力用信号線４０２、インタフェース制御部４０５、バッファ管理部４０６、バッファメモリ４０７、ディスク制御部４０８、ディスク４０９、はそれぞれ、入出力用信号線２０２、インタフェース制御部２０５、バッファ管理部２０６、バッファメモリ２０７、ディスク制御部２０８、ディスク２０９と同等の機能を提供する。高機能処理部４１１は、特定の応用向けの高度機能、例えばソーティ

ング、画像処理、データベースシステムの基本演算、例えば選択処理、写像処理、結合処理、集計処理等を提供する。高機能ＳＡＤインタフェース４１０は、ブロック単位入出力とともに、高機能処理部４１１が提供する高度処理を利用するためのインタフェースを持つ。

【００２１】

図５を用いて、近年提案されている高機能ＮＡＳ型ストレージの構成を説明する。

【００２２】

１つ以上の高機能ＮＡＳ型ストレージ５０３は、ネットワーク５０２によって１つ以上のコンピュータ５０１、５０１’、…と接続される。高機能ＮＡＳ型ストレージ５０３はストレージコントローラ５０４とディスク５０９からなり、ストレージコントローラ５０４はさらにネットワーク制御部５０５、バッファ管理部５０６、バッファメモリ５０７、ディスク制御部５０８、及び高機能処理部４１１からなる。

【００２３】

ネットワーク制御部５０５、ディスク５０９、バッファ管理部５０６、バッファメモリ５０７、ディスク制御部５０８は、それぞれネットワーク制御部３０５、ディスク３０９、バッファ管理部３０６、バッファメモリ３０７、ディスク制御部３０８と同等の機能を持つ。

【００２４】

高機能処理部４１１は、特定の応用向けの高度機能、例えばソーティング、画像処理、データベースシステムの基本演算（例えば、選択処理、写像処理、結合処理、集計処理等）を提供する。高機能ＮＡＳインタフェース５１０は、ブロック単位入出力とともに、高機能処理部４１１が提供する高度処理を利用するためのインタフェースを持つ。また、文献４に記載のシステム及び文献５に記載のシステムでは、高機能処理部４１１が備える機能を外部からダウンロードすることができる。

【００２５】

【発明が解決しようとする課題】

ネットワークに直接結合し、または高機能を提供するストレージを実現する際に、もっとも根本的な課題は、広い応用範囲に対して有効な機能を継続的に提供できるストレージにすることである。応用範囲の広さは、市場の広さにつながり、市場の広さは開発コストの低減と開発スピードの向上につながる。過去に多くのデータベース専用マシンが提案され、ことごとく失敗したが、その原因は応用範囲の狭さにあったと考えられている。応用範囲が狭かったために十分な開発コストがかけられず、そのために汎用品を用いた製品の競争力（開発速度、速度向上等）について行けなかったのである。

【００２６】

広い応用範囲に対して有効な機能を継続的に提供するストレージを実現するためには、機能の拡張性が高いこと、機能の開発コストが低いこと、及び総所有コスト（Ｔｏｔａｌ　Ｃｏｓｔ　ｏｆ　Ｏｗｎｅｒｓｈｉｐ）の観点からコストパフォーマンスが高いこと、の３点が要件である。最も大きくは、これら３点が本発明が解決しようとする課題である。

【００２７】

従来技術では、応用毎の高度機能を必要に応じてストレージにダウンロードすることで拡張性を考慮しているが、開発コストと総所有コストの要件が十分考慮されていない。

【００２８】

文献１に記載のシステムと文献２に記載のシステムでは、ブロックアクセスの上にファイルシステムを作りつけにしており、このため応用範囲が限られている。文献３に記載のシステムは、複数の高度機能を備えることを考察しているが、これら複数の高度機能をどのように作るかについては不明である。文献４は、従来のＯＳに近いソフトウェア階層の上に複数の高度機能を実現する提案である。しかし、応用毎に従来のＯＳ部分にあたるソフトウェアの作りも異なる。例えばＲＤＢＭＳはＯＳのファイルシステムを使わないので、ＲＤＢＭＳにとってはファイルシステムは不要である。すなわち従来のソフトウェア階層をそのままストレ

ージに持ち込んでも、ストレージが担うべき広い応用範囲に対応することが難しい。文献５に記載のシステムはまだ構想段階であるが、ＲＤＢＭＳに焦点をあてており、応用範囲を限定している。

【００２９】

従来技術で十分考慮されていない開発コストと総所有コストの要件についてさらに考察する。

【００３０】

開発コストに関しては、応用毎の高度機能を一つ一つ別個に開発していたのでは、開発コストが増大し、競争力の低下につながる。応用毎の高度機能を実現するプログラム・モジュール（以下モジュールと記す）を低開発コストで開発する必要がある。高度機能を実現するモジュールは応用毎に千差万別であるが、これら高度機能の共通部分をくくりだせれば、共通部分を重複して開発する必要がなくなるとともに共通部分のデバッグも不要になり、低開発コストを実現できる。また、開発したモジュールが高速に実行できる機構が提供できれば、チューニングに要する開発時間が短縮できるため、開発コストが低減できる。

【００３１】

すなわち、開発コストの要件に関しては、以下の課題がある。

【００３２】

（Ａ）複数の応用向け高度機能の共通部分をストレージが提供する。

【００３３】

（Ｂ）該共通部分を低開発コストで実現する。

【００３４】

（Ｃ）該共通部分を用いて複数の応用向け高度機能を実現する。

【００３５】

（Ｄ）複数の応用向け高度機能が共通部分を使う際に保護を行う。

【００３６】

（Ｅ）前記応用向け高度機能が共通部分を使う際に排他制御を行う。

【００３７】

（Ｆ）モジュールを高速に実行できる機構を提供する。

【００３８】

さらに、総所有コストの要件に関しては、ストレージがネットワーク上に多数存在し得ることを鑑みる必要性から、次の課題がある。

【００３９】

（Ｇ）多数のストレージが存在する際に、モジュールを配布する。

【００４０】

がある。

【００４１】

従って、本発明の目的は、以上（Ａ）から（Ｇ）の各課題を解決することにより、広い応用範囲に対して有効な機能を継続的に提供し、次世代のストレージの実現を可能にした拡張型ネットワーク接続二次記憶方法及び装置を提供することである。

【００４２】

【課題を解決するための手段】

以下に上記（Ａ）から（Ｇ）の課題を解決する手段（ａ）から（ｇ）を順に述べる。

【００４３】

（ａ）本発明では、複数の応用の高度機能の共通部分として、応用が用いるデータの塊（オブジェクト）に着目した。オブジェクトの例は、データベースシステムのテーブル、レコード、カラム等のデータ、ファイルシステムではファイル、また、ファイルシステム上の特定のファイルフォーマットを扱う応用ではその特定のファイルフォーマットのファイル、などである。オブジェクトのデータを二次記憶上に配置する方法は、通常、応用または応用分野に対して決まっており、応用毎の高度機能の数に比べずっと少ない。このため、二次記憶上に格納されたオブジェクトにアクセスするモジュール（以下オブジェクトアクセスモジュールと呼ぶ）を高度機能を実現するモジュール（以下機能モジュールと呼ぶ）と分離して、これら２種のモジュール間のインタフェースを明確に定義する。

【００４４】

その上で、様々な応用分野に対応するため、オブジェクトアクセスモジュールを

、コンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供する。オブジェクトアクセスモジュールは、オブジェクトが二次記憶にどのように格納されているかを知っており、このモジュールによって応用毎のオブジェクトを二次記憶の１つ以上のブロックから取り出すことを可能とする。これらにより、様々な応用向けの機能モジュールの開発コストを削減することが可能となる。

【００４５】

なお、本発明が対象とするオブジェクトは、単に連続ブロックに存在する可変長のデータなどのような単純な構造を持つものではない。オブジェクトは、一般に複数の非連続なブロックにまたがって存在し、ブロック間の参照関係も存在する。例えば、Ｕｒｅｓｈ　Ｖａｈａｌｉａ著の文献”ＵＮＩＸ　Ｉｎｔｅｒｎａｌｓ：Ｔｈｅ　Ｎｅｗ　Ｆｒｏｎｔｉｅｒｓ”２６２頁　２６６頁、１９９６年Ｐｒｅｎｔｉｃｅ－Ｈａｌｌ発行；以後文献７と記す）に記載のファイルシステムでは、ひとつのファイルが１段、２段または３段のブロック間参照によって繋がっている。また、データベースのインデックスとして頻繁に用いられるＢ－ｔｒｅｅやＨａｓｈテーブルも、多段のブロック間参照を含む複雑なオブジェクトである。

【００４６】

（ｂ）オブジェクトが二次記憶に格納されている方法（どのブロックのどの部分にどのような順序で格納されているか）のうち宣言的に書ける部分は、宣言的に書くことによって開発コストを削減できる。このため、本発明は、オブジェクトが二次記憶に格納されている方法を宣言的に記述したオブジェクト記述データを導入する。

【００４７】

オブジェクト記述データを、コンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供しつつ、オブジェクト単位のアクセスを低開発コストで実現することを可能とする。

【００４８】

（ｃ）本発明では、機能モジュールを、オブジェクトアクセスモジュールの機能

を用いて実現することによって、開発コストを低減することを可能とする。機能モジュールをコンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供しつつ、応用毎の高度機能を実現する。

【００４９】

（ｄ）オブジェクトアクセスモジュールは共用のモジュールなので、安定したモジュールになりやすい。一方、機能モジュールは応用ごと、高度機能毎のモジュールなので、使用頻度も少なく、バグが取れて安定するのに時間がかかる。本発明はこの点に着目し、機能モジュールからのオブジェクトアクセスモジュールの利用を保護するモジュール（プロテクションモジュール）を導入する。

【００５０】

また、プロテクションモジュールをコンピュータからストレージにダウンロード可能とすることで、保護の方法をオブジェクト毎や応用プログラム毎の拡張を可能とする。

【００５１】

（ｅ）機能モジュールがひとつのオブジェクトを同時に利用することが起こりうるが、その際の排他制御を行う方法を提供する。オブジェクトとオブジェクトの間には、包含関係がある（例えば関係型データベースのテーブルとレコードとカラムは、順に包含関係にある）ため、このような複数のオブジェクトを適切に排他制御するためにオブジェクト間の包含関係を考慮した排他制御を実現したモジュール（ロックモジュール）をストレージが提供する。

【００５２】

ロックモジュールをコンピュータからストレージにダウンロード可能とすることで、排他制御の方法をオブジェクト毎や応用プログラム毎の拡張を可能とする。

【００５３】

（ｆ）モジュールを高速に実行するためにストレージまたは管理マシンにコンパイラを備える。

【００５４】

（ｇ）本発明では、複数のストレージへのモジュールのダウンロードを司る管理マシンを置き、必要に応じてモジュールの加工（代表的にはコンパイル）を行っ

た後ストレージにダウンロードする。

【００５５】

【発明の実施の形態】

本発明の実施の一形態を、図面を参照しながら説明する。なお簡単のため、以下に述べる発明の実施の形態を単に「本実施例」と呼ぶ。

【００５６】

全体構成図１を用いて、本実施例の全体構成を説明する。

【００５７】

図１の全体１０１は、本実施例が好適に用いられるコンピュータシステムであり、ネットワーク１０３と、ネットワーク１０３によって相互接続された１つ以上のコンピュータ１０２、１０２’、…および１つ以上のアクティブネットワークストレージ（ＡＮＳ）１０４からなる。

【００５８】

ネットワーク１０３は、ある団体（企業や学校や類似の団体）の全体や位置部門でよく使用されるＬＡＮでもよく、また地理的に分散した複数の地点を結合するＷＡＮの一部または全部でもよい。またネットワーク１０３は、計算機間結合網や並列計算機内部のプロセッサ要素間の結合網でもよい。また特殊な場合として、ネットワーク１０３がコンピュータとＡＮＳ１０４をつなぐ入出力用信号線（ＳＣＳＩケーブル等）でも差し支えない。

【００５９】

コンピュータ１０２、１０２’、…は、いわゆるパーソナル・コンピュータ、ワークステーション、並列計算機、大型計算機、小型携帯型コンピュータ等、任意のコンピュータでよい。

【００６０】

なお、図１に示したコンピュータ１０２、１０２’、…、ネットワーク１０３、ＡＮＳ１０４の数と構成は、例として示したもので、本発明の範囲を限定するものではない。

【００６１】

ＡＮＳ１０４は、拡張型の二次記憶装置（ストレージ）である。ＡＮＳ１０４は

１つ以上のディスク１１１とアクティブネットワークストレージコントローラ（ＡＮＳＣ）１０５とからなる。

【００６２】

ディスク１１１は電源断後もデータを保持することが可能な記憶媒体（二次記憶）である。ディスク１１１のデータ格納単位には、セクタ、トラック等さまざまな呼称があるが、本実施例では一括してブロックと記す。ディスク１１１は複数のブロックからなり、ブロック単位の入出力を行うことができる。二次記憶がハードディスクであれば、多くの場合ブロックは固定長で５１２バイトないし４Ｋバイトである。メインフレーム計算機用のハードディスクであれば、ブロックは固定長の場合と可変長の場合がある。また、テープドライブ等の二次記憶もブロックはその装置毎に決まっている。なお、ブロック単位でなくバイト単位の入出力インタフェースを提供する二次記憶もあるが、ブロックを１バイトと考えることによって本発明を適用することができる。

【００６３】

ＡＮＳＣ１０５はＡＮＳ１０４の制御を行う部分である。ＡＮＳＣ１０５はさらにネットワーク制御部１０６、モジュール実行部１０７、バッファ管理部１０８、バッファメモリ１０９、ディスク制御部１１０からなる。

【００６４】

ネットワーク制御部１０６は、コンピュータ１０２、１０２’、…をはじめとする外部からネットワーク１０３経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信をネットワーク１０３へ送り出す。モジュール実行部１０７は、ＡＮＳ１０４が提供する各種機能を実現する部分である。各種機能の詳細については、後で詳しく述べる。バッファ管理部１０８は、バッファメモリ１０９の制御を行う部分である。バッファメモリ１０９は、ディスク１１１から得たデータを一時的にためておくメモリである。ディスク制御部１１０は、ディスク１１１にブロックの読み出し・書き込みを行わせる制御を行う。ネットワーク制御部１０６、バッファ管理部１０８、バッファメモリ１０９、ディスク制御部１１０については、従来技術としてよく知られているため、ここではこれ以上詳しく説明しない。

【００６５】

モジュール実行部１０７が提供する各種機能は、大きく分けてブロック単位入出力の機能、オブジェクト単位入出力の機能、応用毎の高度機能、その他の機能の４種類に分類される。

【００６６】

ブロック単位入出力の機能は、従来からストレージがコンピュータに提供している機能である。ブロック読み出し処理（１つ以上のブロックの番号を指定されて、対応するブロックのデータを読み出してコンピュータに返答する処理）と、ブロック書き込み処理（１つ以上のブロック番号と書きこむべきデータを指定されて、対応するブロックにデータを書き込む処理）とが主要な機能である。

【００６７】

オブジェクト単位入出力の機能は、以下に述べる「オブジェクト」の一部または全部を読み出し・書き込みする処理が主要な機能である。オブジェクトとは、コンピュータ１０２、１０２’、…上で動作する応用プログラムがストレージに保存する意味的なデータの塊を指す。オブジェクトの定義は、応用毎または応用分野毎に異なる。オブジェクトの例としては、データベースシステムのテーブル、レコード、カラム等のデータ、ファイルシステムではファイル、また、ファイルシステム上の特定のファイルフォーマットを扱う応用ではその特定のファイルフォーマットのファイル、などがある。

【００６８】

応用毎の高度機能は、応用プログラムが必要に応じて定義した入出力処理である。例えばソーティング、画像処理、データベースシステムの基本演算、例えば選択処理、写像処理、結合処理、集計処理等が高度機能の例として考えられる。

【００６９】

その他の機能としては、ディスク１１１の初期化、バックアップ、複製等の機能が考えられる。

【００７０】

モジュール実行部１０７は、上記機能を１つ以上のプログラム・モジュールであるモジュール１１２、１１２’、１１２”、…の組み合わせによって実現する。

モジュール１１２、１１２’、１１２”、…は、提供する機能によって機能モジュール１１３、オブジェクトアクセスモジュール１１４、ブロックアクセスモジュール１１５、プロテクションモジュール１１６、ロックモジュール１１７の５種類に分類される。さらに新たな種類のモジュールを定義しても差し支えない。

【００７１】

機能モジュール１１３は、上記応用毎の高度機能を実現するモジュールである。高度機能の種類によって複数存在しうる。オブジェクトアクセスモジュール１１４は、上記オブジェクト単位入出力を実現するモジュールである。オブジェクトの種類によって複数存在しうる。ブロックアクセスモジュール１１５は、上記ブロック単位入出力を実現するモジュールである。ブロックの種類によって複数存在しうる。プロテクションモジュール１１６は、モジュール間の呼び出しを選択的に許可または禁止することによって、保護を実現するモジュールである。保護の方法によって複数存在しうる。ロックモジュール１１７は、オブジェクト単位の排他制御を実現するモジュールである。排他制御の方法によって複数存在しうる。

【００７２】

ＡＮＳインタフェース１１８は、ＡＮＳ１０４とコンピュータ１０２、１０２’、…をはじめとする外部とのインタフェースである。応用毎の高度機能、オブジェクト単位入出力、ブロック単位入出力、保護を行うため、モジュール登録・削除１１９、メソッド起動１２０、オブジェクト記述データ登録・削除１２１、保護ポリシー登録・削除１２２を含む。

【００７３】

モジュール登録・削除１１９は、上記モジュール１１２、１１２’、１１２”、…を外部からＡＮＳ１０４に登録、またはＡＮＳ１０４から削除するインタフェースである。メソッド起動１２０は、ＡＮＳ１０４に登録されたモジュールの特定の機能を起動し、応答を受けるインタフェースである。なおメソッドとは、後述するように、モジュールが提供する手続きである。オブジェクト記述データ登録・削除１２１は、オブジェクトが二次記憶にどのように格納されているかを記述したデータであるオブジェクト記述データを、ＡＮＳ１０４に登録、またはＡ

ＮＳ１０４から削除するインタフェースである。オブジェクト記述データに関しては、後で詳しく述べる。保護ポリシー登録・削除１２２は、プロテクションモジュール１１６が実現するモジュール間の保護の方法を記述したデータである保護ポリシーを、ＡＮＳ１０４に登録、またはＡＮＳ１０４から削除するインタフェースである。保護ポリシーに関しては、後で詳しく述べる。

【００７４】

以上が本実施例の全体構成である。

【００７５】

図６を用いて、ＡＮＳ１０４の内部構成および代表的動作を説明する。

【００７６】

モジュール登録・削除１１９、メソッド起動１２０、オブジェクト記述データ登録・削除１２１、保護ポリシー登録・削除１２２のいずれかの要求がネットワークまたは入出力用信号線からＡＮＳ１０４に送られると、該要求はネットワーク制御部１０６が受け取る（６０１）。

【００７７】

ネットワーク制御部１０６は、該要求を、その種類に応じてモジュール実行部１０７か、バッファ管理部１０８に送る（６０２、６１５）。バッファ管理部１０８に送られる要求は、メソッド起動１２０のうち、ブロック単位入出力等、従来からＳＡＤ型ストレージが提供している機能である。モジュール実行部１０７に送られる要求は、これ以外のすべての要求である。なわちモジュール登録・削除１１９、オブジェクト記述データ登録・削除１２１、保護ポリシー登録・削除１２２、及びブロック単位入出力以外のメソッド起動１２０がモジュール実行部１０７に送られる。

【００７８】

モジュール実行部１０７の内部には、各種モジュールの管理・実行を行うモジュール管理部６５０がある。モジュール登録・削除１１９はモジュール管理部６５０によってディスパッチされ、適切なモジュールに配送される。

【００７９】

応用毎の高度機能に関するメソッド起動要求は機能モジュール６５１に送られて

処理される（６０３）。オブジェクト単位入出力に関するメソッド起動要求およびオブジェクト記述データ登録・削除１２１は、オブジェクトアクセスモジュール６５２に送られて処理される（６０９）。ブロック単位入出力に関するメソッド起動要求で、バッファ管理部が直接扱えない要求はブロックアクセスモジュール６５３に送られる（６１１）。保護ポリシー登録・削除１２２は、プロテクションモジュール６５４に送られて処理される（６０９）。モジュール登録・削除１１９は、モジュール管理部６５０が直接処理する。すなわち、モジュールの登録の場合には送られてきたモジュールを新たなモジュールとしてモジュール実行部１０７内に保存する。また、削除の場合には指定されたモジュールをモジュール実行部１０７内から削除する。

【００８０】

機能モジュール６５１は、オブジェクトアクセスモジュール６５２が提供するオブジェクト単位入出力の機能を利用し、またロックモジュール６５５が提供する排他制御の機能を利用して（６０５）、高度機能を実現する。ロックモジュール６５５は、さらに、オブジェクトアクセスモジュール６５２の機能を利用して（６０８）排他制御を実現する。

【００８１】

機能モジュール６５１がオブジェクトアクセスモジュール６５２の機能を利用する際には、機能モジュール６５１がオブジェクトアクセスモジュール６５２のメソッドを起動する。このメソッド起動は、モジュール管理部６５０によってまずプロテクションモジュール６５４に送られて起動の可否が判定され（６０４）、起動可ならオブジェクトアクセスモジュール６５２にメソッド起動要求が送られる（６０７）。

【００８２】

オブジェクトアクセスモジュール６５２は、ブロックアクセスモジュール６５３の機能であるブロック単位入出力を利用して（６１０）、オブジェクト単位入出力を実現する。

【００８３】

ブロックアクセスモジュール６５３は、バッファ管理部１０８にブロック取得要

求を送るか（６１３）、または直接バッファメモリ１０９を参照・変更することによって（６１４）、ブロック単位入出力を実現する。バッファ管理部１０８はまた、モジュール管理部６５０がモジュールを保存する際等に直接起動されうる（６１２）。

【００８４】

バッファ管理部１０８は、バッファメモリ１０９の内容を管理する（６１６）。例えば、バッファメモリ１０９のブロックのリプレースメント、ダーティなブロックのディスク１１１への書き戻し処理の起動、現在バッファメモリ１０９に存在しないブロックに対するブロック参照要求をディスク制御部１１０に転送し（６１７）、ブロックをディスク制御部１１０からバッファメモリ１０９へ転送する（６１８）の起動を行う。

【００８５】

ディスク制御部１１０はディスク１１１へのブロック単位入出力要求を受け、実際にディスク１１１を駆動して入出力を行う（６１９）。

【００８６】

以上が、ＡＮＳ１０４の内部構成および代表的動作である。各モジュールの内部構造および動作、オブジェクト記述データ、保護ポリシーについては、さらに以下に詳しく述べる。

【００８７】

図７を用いて、各モジュール共通の内部構成および代表的動作を説明する。

【００８８】

オブジェクトアクセスモジュール１１４、ブロックアクセスモジュール１１５、プロテクションモジュール１１６、ロックモジュール１１７はいずれも、図７に示す構造を共通に持つ。

【００８９】

モジュール７０１には１つ以上の手続き（メソッド７０２）と、０個以上の変数（属性７０３）がある。モジュール７０１の他のモジュールまたは外部からメソッド起動７０４を受けると、対応するメソッドが開始される。メソッド７０２は、自モジュールの属性７０３を参照・変更し、または他モジュールのメソッド起

動７０５によって処理を進める。

【００９０】

機能モジュールの場合、メソッドとして、応用毎の高度機能を外部および他モジュールに提供する。機能モジュールが用いるメソッドは、オブジェクトアクセスモジュールの各メソッド及びロックモジュールの各メソッドである。

【００９１】

オブジェクトアクセスモジュールの場合、メソッドとしてオブジェクト（とその部分）を指定してその内容を参照・変更する手続きを、外部及び他モジュールに提供する。

【００９２】

より具体的に、オブジェクトの部分を参照するメソッドとして、ｇｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ）（オブジェクトを指定して該オブジェクトの内容全部を参照する）、ｇｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｏｆｆｓｅｔ、ｓｉｚｅ）（オブジェクトとオブジェクト先頭からのバイト数、対象のバイト数を指定して、指定したデータを参照する）、ｇｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｔａｇ１、ｔａｇ２、...）（オブジェクトと、取り出すべき部分を指定するタグとを指定して、指定したデータを参照する）ｇｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｉ、ｊ、ｋ、...）（オブジェクトと、取り出すべき部分を指定するインデックスとを指定して、指定したデータを参照する）ｇｅｔＮｅｘｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ）（オブジェクトの次の部分を参照する）を提供する。なお、タグについては、図１０、図１２、図１４を用いて後で説明する。

【００９３】

また、オブジェクトを変更するメソッドとして、ｓｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｄａｔａ）（オブジェクトを指定して該オブジェクトの内容全部を変更する）、ｓｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｄａｔａ、ｏｆｆｓｅｔ、ｓｉｚｅ）（オブジェクトとオブジェクト先頭からのバイト数、対象のバイト数を指定して、指定したデータを変更する）、ｓｅｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｄａｔａ、ｔａｇ１、ｔａｇ２、...）（オブジェクトと、変更すべき部分を指定するタグとを指定して、指定したデータを変更する）、ｓｅｔＯｂｊｅｃｔ（Ｏｂｊ

ｅｃｔ、ｄａｔａ、ｉ、ｊ、ｋ、...）（オブジェクトと、変更すべき部分を指定するインデックスとを指定して、指定したデータを変更する）ｓｅｔＮｅｘｔＯｂｊｅｃｔ（Ｏｂｊｅｃｔ、ｄａｔａ）（オブジェクトの次の部分を変更する）を提供する。これらのメソッドの実現には、ブロックアクセスモジュールのメソッドを用いる。

【００９４】

上記の各メソッドに加えて、該オブジェクトアクセスモジュールがオブジェクト記述データを用いる場合、後述のようにオブジェクト記述データを登録・削除するメソッドａｄｄＯｂｊｅｃｔＤｅｓｃｒｉｐｔｉｏｎ（ｄｅｓｃ）と　ｄｅｌｅｔｅＯｂｊｅｃｔＤｅｓｃｒｉｐｔｉｏｎ（ｄｅｓｃ）を提供する。

【００９５】

ブロックアクセスモジュールの場合、メソッドとして、ブロックを参照する手続きとブロックを変更する手続きとを、外部および他モジュールに提供する。より具体的には、ｇｅｔＢｌｏｃｋ（ｂｌｏｃｋＩＤ）（ブロック番号を指定してブロックを参照する）と　ｓｅｔＢｌｏｃｋ（ｂｌｏｃｋＩＤ、ｄａｔａ）（ブロック番号を指定してブロックを変更する）とを提供する。

【００９６】

プロテクションモジュールの場合、メソッド起動の仕様（送信元、送信先、メソッド名）を与えられて該メソッド起動の可否を判断するｇｅｔＰｒｏｔｅｃｔｉｏｎ（ｓｏｕｒｃｅ、ｄｅｓｔ、ｍｅｔｈｏｄ）をモジュール管理部に提供する。これに加えて、該プロテクションモジュールが保護ポリシーを登録・削除できる場合、後述のように保護ポリシー記述データを登録・削除するメソッドａｄｄＰｒｏｔｅｃｔｉｏｎＰｏｌｉｃｙ（ｐｏｌｉｃｙ）とｄｅｌｅｔｅＰｒｏｔｅｃｔｉｏｎＰｏｌｉｃｙ（ｐｏｌｉｃｙ）を提供する。

【００９７】

ロックモジュールの場合、オブジェクトにロックをかけるメソッドｌｏｃｋＯｂｊｅｃｔ（ｏｂｊｅｃｔ、ｌｏｃｋ＿ｍｏｄｅ）、オブジェクトのロックをはずすメソッドｕｎｌｏｃｋＯｂｊｅｃｔ（ｏｂｊｅｃｔ、ｌｏｃｋ＿ｍｏｄｅ）、オブジェクト間の包含関係を登録・削除するメソッドとして、ａｄｄＯｂｊｅｃ

ｔＲｅｌａｔｉｏｎｓｈｉｐ（ｏｂｊｅｃｔ、ｏｂｊｅｃｔ）とｄｅｌｅｔｅＯｂｊｅｃｔＲｅｌａｔｉｏｎｓｈｉｐ（ｏｂｊｅｃｔ、ｏｂｊｅｃｔ）を、外部及び他モジュールに提供する。

【００９８】

次に、図８を用いて、オブジェクトアクセスモジュールの中で、特にオブジェクト記述データを用いるモジュールの内部構成および代表的動作を説明する。

【００９９】

オブジェクト記述データを用いるオブジェクトアクセスモジュールも、モジュール８０１内に１つ以上のメソッド８０２と０個以上の属性８０３があって、メソッド起動８０４によってメソッド８０２が起動され、属性８０３を参照・変更し、または他モジュールのメソッド起動８０５によって処理を進める。オブジェクト記述データを用いるオブジェクトアクセスモジュールはさらに、オブジェクト記述データ８０６をメソッド８０２内に保持し、オブジェクト記述データ登録・削除８０７に応じて、オブジェクト記述データ８０６を登録または削除する。オブジェクト記述データ８０６の構成及び使用方法については、図１０、図１２、図１４を用いて後で説明する。

【０１００】

次に、図９を用いて、プロテクションモジュールの内部構成および代表的動作を説明する。

【０１０１】

プロテクションモジュールは、モジュール９０１内に保護判定部９０２と保護ポリシー９０３があって、保護判定メソッド起動９０４によって保護判定部９０２が起動され、保護ポリシー９０３を参照することによってメソッド起動の可否を判定する。プロテクションモジュールはさらに、保護ポリシー登録・削除９０５に応じて、保護ポリシー９０３を登録または削除する。

【０１０２】

次に、オブジェクト記述データについて説明する。本実施例のオブジェクト記述データには３種類の異なる形式がある。第１の形式は表形式で、ブロック内、ブロック間のデータの並びが固定である場合（またはそのようなオブジェクトの部

分）によい形式である。例えば、ファイルシステムのファイルや、データベースシステムのインデックスなどにはこの形式が有利である。

【０１０３】

第２の形式はパーザ形式で、ブロック内、ブロック間のデータの並びを文脈自由文法で記述する方法である。この形式は、ブロック内、ブロック間のデータの並びが比較的自由度を持っている場合（またはそのようなオブジェクトの部分）によい形式である。例えばデータベースシステムの表や、ファイルシステムのディレクトリなどにはこの形式が有利である。

【０１０４】

第３の形式はパターンマッチ形式で、ブロックまたはオブジェクトの特定部分のデータが特定の値かパターンをあるかどうかによってファイルフォーマットを特定する方法である。この形式は、同じ構造のオブジェクトに異なるフォーマットのデータが格納されている可能性がある場合によい形式である。例えばファイルシステムのファイル中に複数のファイルフォーマットの画像ファイルが格納されており、ファイルフォーマットに応じて高度機能を切り替えたい場合などにはこの形式が有利である。

【０１０５】

図１０を用いて、表形式のオブジェクト記述データの構成を説明する。

【０１０６】

オブジェクト記述データ（表形式）１００１は、タグ１００２、タイプ１００３、オフセット１００４、サイズ１００５、カウント１００６、ブロックタイプ１００７の６個の要素からなる表である。この表によって、ブロック内及びブロック間のデータの並びを得る。１つの行がブロック内の意味的な一塊のデータを表す。

【０１０７】

タグ１００２は、一塊のデータにつけた名前である。この名前は、このデータ部分にアクセスする際に用いられる。タイプ１００３は、該データの型を示す。オフセット１００４は、該データがオブジェクトの先頭から何バイト目から格納されているかを示す。サイズ１００５は、該データのバイト数を示す。カウント１

００６は、該データが連続して並ぶ場合の総数を表す。ブロックタイプ１００７は、該データが他のブロックを参照するブロック番号である場合に、参照先のブロック番号の形式を指定する。

【０１０８】

図１１を用いて、表形式オブジェクト記述データの利用例を説明する。

【０１０９】

この例では、文献６に記載のファイルシステムのファイルを一連のオブジェクトとして記述する。該ファイルシステムのファイルは、５種類の異なる形式のブロックによって構成されている。第１に、ＵＦＳ＿ｉｎｏｄｅオブジェクト１１０１は、該ファイルのメタデータを格納するオブジェクトである。ｄｉ＿ｍｏｄｅ、ｄｉ＿ｎｌｉｎｋ、ｄｉ＿ｕｉｄ、ｄｉ＿ｇｉｄ、ｄｉ＿ｓｉｚｅ、ｄｉ＿ａｄｄｒ１、ｄｉ＿ａｄｄｒ２、ｄｉ＿ａｄｄｒ３、ｄｉ＿ａｄｄｒ４、ｄｉ＿ｇｅｎ、ｄｉ＿ａｔｉｍｅ、ｄｉ＿ｍｔｉｍｅ、ｄｉ＿ｃｔｉｍｅの各要素からなることを示している。例えばｄｉ＿ｍｏｄｅに着目すると、型はｓｈｏｒｔ型で、オブジェクトの先頭０バイト目から２バイトを占め、繰り返し数は１回である。また、別の例としてｄｉ＿ａｄｄｒ１に着目すると、型はＢＬＯＣＫ型（すなわち他のブロックを参照するブロック番号）で、オブジェクトの先頭１２バイト目から３バイトを占め、繰り返し数は１０回であり、参照先のブロックはＵＦＳ＿ｄａｔａオブジェクト１１０２である。

【０１１０】

同様に、ＵＦＳ＿ｄａｔａオブジェクト１１０２、ＵＦＳ＿ｉｎｄｉｒｅｃｔ１オブジェクト１１０３、ＵＦＳ＿ｉｎｄｉｒｅｃｔ２オブジェクト１１０４、ＵＦＳ＿ｉｎｄｉｒｅｃｔ３オブジェクト１１０５が記述されており、全体で０段ないし３段の間接参照を含むファイルの構造が定義されている。

【０１１１】

図１２を用いて、パーザ形式オブジェクト記述データの構成を説明する。

【０１１２】

オブジェクト記述データ（パーザ形式）１２０１は、タグ１２０２、初期化コード１２０３、コンテクスト１２０４、コード１２０５の各要素を持つ表である。

この表から、ブロックをパーズするパーザを構築し、ブロックをパーズすることによりブロック内及びブロック間のデータの並びを得る。

【０１１３】

タグ１２０２は、一塊のデータにつけた名前である。初期化コード１２０３は、該データをパーズする際にあらかじめ実行すべき命令列である。コンテクスト１２０４は、パーズする文脈を指定する。本実施例の指定方法では、文脈自由文法と若干の拡張を施した記述が可能である。コード１２０５は、文脈をパーズした後に実行するべき命令列である。

【０１１４】

図１３を用いて、パーザ形式オブジェクト記述データの利用例を説明する。

【０１１５】

テーブル定義１３０１は、データベースシステムにおける表定義の例を示している。ｃ＿ｉｄ、ｃ＿ｎａｍｅ、ｃ＿ａｄｄｒｅｓｓの３つのカラムを持つｃｕｓｔｏｍｅｒテーブルを定義している。

【０１１６】

文法１３０２は、テーブル定義１３０１に対応するｃｕｓｔｏｍｅｒテーブルのオブジェクト記述データである。全部で５行の表であり、表全体をパーズするｃｕｓｔｏｍｅｒ（）、行をパーズするｒｅｃｏｒｄ（）、ｃ＿ｉｄをパーズするｃ＿ｉｄ（）、ｃ＿ｎａｍｅをパーズするｃ＿ｎａｍｅ（）、ｃ＿ａｄｄｒｅｓｓをパーズするｃ＿ａｄｄｒｅｓｓ（）がそれぞれの行に定義されている。例えば、ｒｅｃｏｒｄ（）に着目すると、「１つの行は先頭にレコードサイズをあらわすｌｏｎｇ型のデータ、続いてｃ＿ｉｄ、ｃ＿ｎａｍｅ、ｃ＿ａｄｄｒｅｓｓが続く」ことがコンテクスト部に示されている。また、ｒｅｃｏｒｄをパーズし終わった際にｆｏｕｎｄＯｂｊｅｃｔ手続きを呼ぶことによって、行を検出したことを宣言する処理がコード部に示されている。このテーブル定義１３０１により、ｃｕｓｔｏｍｅｒテーブルの各行、各列を検出することができる。なお、この例では用いなかったが、コード１２０５中でｓｗｉｔｃｈＢｌｏｃｋ（ｂｌｏｃｋＩＤ）手続きを用いることにより、処理対象ブロックを移動することも可能である。

【０１１７】

図１４を用いて、パターンマッチ形式オブジェクト記述データの構成を説明する。

【０１１８】

オブジェクト記述データ（パターンマッチ形式）１４０１は、ファイルフォーマット１４０２、タグ１４０３、パターン１４０４の要素からなる表である。１つのファイルフォーマット１４０２に対しタグ１４０３とパターン１４０４との組が１つ以上対応する。ファイルフォーマット１４０２はファイルフォーマットの名前である。タグ１４０３は他のオブジェクト記述データによって与えられる、オブジェクトの一部をなす一塊のデータの名前である。パターン１４０４は、タグ１４０３で示されたデータとパターンマッチを行うパターンである。

【０１１９】

ファイルフォーマット１４０２に対応付けられたタグ１４０３とパターン１４０４との組がすべてマッチした場合に、オブジェクトをファイルフォーマット１４０２に示されたファイルフォーマットであると判定する。

【０１２０】

例えば、１４の１行目および２行目を見ると、あるオブジェクトのｈｅａｄｅｒ１というタグをつけたデータが”#!”であって、ｈｅａｄｅｒ２というタグをつけたデータが”／ｂｉｎ／ｓｈ”である場合に、このオブジェクトをｓｈｅｌｌ＿ｓｃｒｉｐｔ（シェルスクリプト）であると判定することができる。

【０１２１】

図１５を用いて、保護ポリシーの構成を説明する。

【０１２２】

保護ポリシーは、プロテクションモジュールが保持するデータで、モジュール間の呼び出しを選択的に許可または禁止するために用いる。保護ポリシー記述データ１５０１は、送信元１５０２、送信先１５０３、メソッド１５０４、許可・禁止１５０５からなる。

【０１２３】

送信元１５０２は、メソッド起動の送信元、送信先１５０３は、メソッド起動の

送信先、メソッド１５０４は、起動対象のメソッド名、許可・禁止１５０５は、「許可」または「禁止」である。プロテクションモジュールは、メソッド起動に際して、送信元、送信先、メソッドを保護ポリシー記述データ１５０１の送信元１５０２、送信先１５０３、メソッド１５０４と照合し、許可・禁止１５０５の値にしたがって、該メソッド起動を許可するか禁止するかを決定する。

【０１２４】

図１６を用いて、表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の手順を説明する。

【０１２５】

オブジェクト単位入出力処理には、上述のようにいくつかのバリエーションがあるが、その代表として、ｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、”ｔａｇ１”、”ｔａｇ２”...）の操作を表形式のオブジェクト記述データを用いて実現する際の手順を説明する。表形式のオブジェクト記述データを用いて実現されるｇｅｔＯｂｊｅｃｔの他のバリエーションも、同様の手順で実現できる。ｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、”ｔａｇ１”、”ｔａｇ２”...、”ｔａｇＮ”）は、オブジェクトの先頭ブロックのｔａｇ１が指定するブロックを参照し、参照先ブロックのｔａｇ２が指定するブロックを参照し、という要領でブロックを渡り歩いていき、最後にｔａｇＮに合致するデータ部分を参照するメソッドである。

【０１２６】

ステップ１６０１からステップ１６０７はループである。ステップ１６０１で、最初のタグに合致するオブジェクト記述データの行を検索する。ステップ１６０２で第２タグがまだあれば（Ｙ）、ステップ１６０３に制御を移し、なければステップ１６０８に制御を移す。ステップ１６０３で、検索結果行のタイプ１００３がＢＬＯＣＫ型かを検査する。結果が真（Ｙ）なら、ステップ１６０４へ制御を移し、偽（Ｎ）なら異常終了する。ステップ１６０４で、処理対象ブロック（ループの最初の場合にはオブジェクトの先頭ブロック）のオフセット１００４からサイズ１００５分のデータを参照する。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。ステップ１６０５で、ステップ１６０４で得たデータを次の処理対象ブロックのブロック番

号として保持する。ステップ１６０６で、使用するオブジェクト記述データを、ブロックタイプ１００７用のオブジェクト記述データに切り替える。ステップ１６０７で、第１タグを捨て、第２タグ以降を第１タグから始まるように左シフトする。ステップ１６０７の後は、ステップ１６０１に戻る。ステップ１６０８では、処理対象ブロックのオフセット１００４からサイズ１００５分のデータを参照し、結果をｇｅｔＯｂｊｅｃｔ（）の返り値として呼び出し元に返す。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。

【０１２７】

以上が表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の流れである。

【０１２８】

図１７を用いて、パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の手順を説明する。この説明では、ｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、ｉ、ｊ、ｋ、...）の手順を説明する。

【０１２９】

ステップ１７０１で、引数列ｉ、ｊ、ｋ、…をターゲット列として保持する。ステップ１７０２で、オブジェクトのパーザを起動し、処理対象ブロック（最初の場合オブジェクトの先頭ブロック、パーズ処理の進行によって他のブロックに処理対象が移動する場合もある）。パーザはターゲット列を検出するか、オブジェクトの最後までパーズが終了すると、ステップ１７０２を完了する。ステップ１７０３で、パーザがターゲット列を検出してパーズが成功したかを検査する。真（Ｙ）なら、ステップ１７０４でパーズ結果をｇｅｔＯｂｊｅｃｔ（）の返り値として呼び出し元に返す。偽（Ｎ）なら、異常終了する。

【０１３０】

以上がパーザ形式のオブジェクト記述データを用いるｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、ｉ、ｊ、ｋ、...）処理の流れである。

【０１３１】

図１８を用いて、パーザ形式のオブジェクト記述データを用いるオブジェクト単

位入出力処理の手順を説明する。この説明では、ｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、”ｔａｇ”）の手順を説明する。

【０１３２】

ステップ１８０１で、ｔａｇをターゲットタグとして保持する。ステップ１８０２で、オブジェクトのパーザを起動し、処理対象ブロック（最初の場合オブジェクトの先頭ブロック、パーズ処理の進行によって他のブロックに処理対象が移動する場合もある）。パーザはターゲット列を検出するか、オブジェクトの最後までパーズが終了すると、ステップ１８０２を完了する。ステップ１８０３で、パーザがターゲット列を検出してパーズが成功したかを検査する。結果が真（Ｙ）なら、ステップ１８０４でパーズ結果をｇｅｔＯｂｊｅｃｔ（）の返り値として呼び出し元に返す。結果が偽（Ｎ）なら、異常終了する。

【０１３３】

以上がパーザ形式のオブジェクト記述データを用いるｇｅｔＯｂｊｅｃｔ（ｏｂｊｅｃｔ、”ｔａｇ”）処理の流れである。

【０１３４】

図１９を用いて、パーザ形式のオブジェクト記述データ中でオブジェクトの部分のパーズに成功した際に起動されるｆｏｕｎｄＯｂｊｅｃｔ（”ｔａｇ”、ｏｆｆｓｅｔ、ｓｉｚｅ、ｉ、ｊ、ｋ、...）の処理手順を説明する。ｆｏｕｎｄＯｂｊｅｃｔ（）と、図１７のｇｅｔＯｂｊｅｃｔ（）または図１８のｇｅｔＯｂｊｅｃｔ（）との組み合わせにより、パーザ形式のオブジェクト記述データによるオブジェクト単位入出力が実現される。

【０１３５】

ステップ１９０１で、ｇｅｔＯｂｊｅｃｔ（）が保持するターゲットタグと引数のｔａｇとが合致するかを検査する。結果が真（Ｙ）なら、ステップ１９０３に制御を移し、結果が偽（Ｎ）ならステップ１９０２に制御を移す。ステップ１９０２で、ｇｅｔＯｂｊｅｃｔ（）が保持するターゲット列に、引数ｉ、ｊ、ｋ...が合致するかを検査する。結果が真（Ｙ）なら、ステップ１９０３に制御を移し、結果が偽（Ｎ）なら何もせずに終了する。ステップ１９０３で、ｇｅｔＯｂｊｅｃｔ（）の処理対象ブロックのｏｆｆｓｅｔからｓｉｚｅ分のデータを参照

し、パーズ結果として保持する。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。

【０１３６】

以上がパーザ内で用いられるｆｏｕｎｄＯｂｊｅｃｔ処理の流れである。以上のように、オブジェクト単位入出力、高機能入出力が、ＡＮＳにダウンロードされるモジュールの組み合わせによって実現できる。オブジェクト単位入出力には、オブジェクトの構造を宣言的に記述したオブジェクト記述データが利用できる。これらにより、モジュールの開発コストを低く押さえることができる。

【０１３７】

図２０を用いて、ＡＮＳ利用コンピュータからＡＮＳへのモジュール登録動作の構成を説明する。

【０１３８】

ＡＮＳ利用マシン２００１は、ネットワーク２００３を通じてモジュール２００６をＡＮＳ　２００２に登録する。ＡＮＳ　２００２は前述のように、ＡＮＳＣ２００４とディスク２００５とからなる。この場合モジュール２００６は、ＡＮＳ　２００２の機種に左右されないために、プラットフォーム独立な言語で記述されるのが、必須ではないが望ましい。プラットフォーム独立な言語としては、インタプリタ型言語をはじめとしていくつかの選択肢がありうる。特に、ネットワーク上での移動を意識して、プラットフォーム独立な実行プログラム、安全な型システム、及び安全な実行時環境を提供する言語が知られており、モジュール２００６の記述に好適である。

【０１３９】

しかし、このようなネットワーク上での移動を意識した言語はしばしば、実行時性能が通常のコンパイル型言語に比べて劣る恐れがある。この問題の解決には、２つの解決策が考えられる。第１の選択肢は、ＡＮＳ内部にコンパイラを持ち、ネットワーク上での移動を意識した言語をより高速な実行プログラム、例えばマシン語で記述された実行プログラムにコンパイルしなおす、という選択肢である。この場合、コンパイルはモジュール管理部６５０で行うことができる。

【０１４０】

また、第２の選択肢は、ＡＮＳの外部に、複数機種のＡＮＳ向けのコンパイラを持つコンピュータを用意し、このコンピュータ（ＡＮＳ管理コンピュータ）経由でモジュールをＡＮＳへ登録する、という選択肢である。この選択肢は、価格性能比等の理由から、ＡＮＳにコンパイラのような大規模ソフトウェアを搭載するのが困難である場合や、ＡＮＳがネットワーク上に多数存在する場合に有効である。

【０１４１】

図２１を用いて、ＡＮＳ利用コンピュータからＡＮＳへの、ＡＮＳ管理コンピュータを経由したモジュール登録動作の構成を説明する。

【０１４２】

ＡＮＳ利用コンピュータ２１０１は、ネットワーク２１０３を通じてプラットフォーム独立モジュール２１０７をＡＮＳ管理コンピュータ２１０４に送る。ＡＮＳ管理コンピュータ２１０４のＡＮＳ管理部２１１０は、プラットフォーム独立モジュール２１０７の送付先ＡＮＳに応じて後述するＡＮＳ管理表２１１１を参照し、モジュールコンパイラ２１０９、２１０９’、…の中から使用すべきコンパイラを選択する。そして、プラットフォーム独立モジュール２１０７を該コンパイラでコンパイルし、コンパイル済みモジュール２１０８を得る。さらにＡＮＳ管理部２１１０は、ＡＮＳ管理表２１１１によってコンパイル済みモジュール２１０８の送付先ＡＮＳのネットワークアドレスを得、コンパイル済みモジュール２１０８を該送付先ＡＮＳに送付する。送付先ＡＮＳ　２１０２は、前述のとおりＡＮＳＣ　２１０５とディスク２１０６とからなる。

【０１４３】

図２２を用いて、ＡＮＳ管理表の構成を説明する。

【０１４４】

ＡＮＳ管理テーブル２２０１は、ＡＮＳ名２２０２、ネットワークアドレス２２０３、機種２２０４、コンパイラ２２０５の各要素からなる。複数のＡＮＳのネットワークアドレス、機種、モジュールコンパイラの対応関係を保持する。１行が１つのＡＮＳに対応する。

【０１４５】

ＡＮＳ名２２０２は、ＡＮＳの名前である。ネットワークアドレス２２０３は、該　ＡＮＳのネットワークアドレスである。機種２２０４は、該　ＡＮＳの機種である。コンパイラ２２０５は、該　ＡＮＳのモジュールをコンパイルするために使用するモジュールコンパイラの名前である。

【０１４６】

以上がＡＮＳ管理表の構成の説明である。

【０１４７】

ＡＮＳ管理部２１１０、ＡＮＳ管理表２１１１、モジュールコンパイラ２１０９、２１０９’、…を持つＡＮＳ管理コンピュータ２１０４によって、モジュールを高速実行し、かつコンパイルに関する管理コストの低いＡＮＳが実現できる。

【０１４８】

ＡＮＳの応用例として、ＡＮＳ利用コンピュータから複数のＡＮＳへストライプ入出力を実現する方法を、図２３を用いて説明する。

【０１４９】

ＡＮＳ利用コンピュータ２３０１がファイルへの入出力を行う場合、該ファイルを複数のディスクに存在させておくストライピイングを行うことによって、複数のディスク入出力を複数のディスクにばらまくことができ、高速な入出力が可能となる。これを以下の手順で実現する。

【０１５０】

ＡＮＳ利用コンピュータ２３０１が複数のＡＮＳ　２３０４にストライプされたファイルを読み書きする場合、ネットワーク２３０２を通じてストライプ管理ＡＮＳ　２３０３へストライプ構成情報取得要求２３０５を送る。この際、ストライプ管理ＡＮＳ　２３０３は、該ファイルがＡＮＳ　２３０４にどのようにストライプされているかに関する情報（ファイルのどのブロックがどのＡＮＳに格納されているかを示した情報）であるストライプ構成情報を返答する。この機能は、ストライプ管理ＡＮＳ　２３０３のオブジェクトアクセスモジュール１１４として実現される。

【０１５１】

ＡＮＳ利用コンピュータ２３０１は次に、該ストライプ構成情報から、該ファイルのうち読み書きを行いたい部分を格納しているＡＮＳを計算し、該当する１つ以上のＡＮＳに対し、入出力要求２３０６を送る。この２段階の操作により、複数のＡＮＳに対するストライプ入出力が実現される。

【０１５２】

ストライプは従来サーバコンピュータが実現し、ストレージはサーバからの入出力要求を受けるだけの受け身の存在であったが、上記の構成方法により、ストレージ自身がストライプを実現することができる。これは、ストレージの構成変更容易性または拡張容易性につながる。例えば、１台のストレージを購入したユーザが、ストレージのボトルネックを解消するために１台のストレージを追加購入し、ストライプ入出力を実現する場合、従来のサーバによるストライプの実現方法では、同時にサーバも新たに購入する必要があり、ストレージの構成変更に多大なコストがかかっていた。しかし、上記の方法によれば、追加のサーバは必要なく、単に２台のストレージを用意してオブジェクトアクセスモジュール１１４をダウンロードすれば、ストライプが実現できるという利点がある。

【０１５３】

【発明の効果】

ネットワークに直接結合し、または高機能を提供するストレージを実現するため

（ａ）応用が用いるデータの塊（オブジェクト）を入出力する機能であるオブジェクトアクセスモジュールを、複数の応用向けの高度機能の共通部分としてストレージが提供し、

（ｂ）オブジェクトが二次記憶に格納されている方法を宣言的に記述するオブジェクト記述データを提供し、

（ｃ）機能モジュールとオブジェクトアクセスモジュールを分離することにより、共通部分であるオブジェクト入出力機能を用いて複数の応用向けの高度機能を実現することを可能にし、

（ｄ）機能モジュールからのオブジェクトアクセスモジュールの利用を保護する

モジュール（プロテクションモジュール）を提供し、

（ｅ）複数のオブジェクトを適切に排他制御するために、オブジェクト間の包含関係を考慮した排他制御を実現したロックモジュールを提供し、

（ｆ）ストレージまたは管理マシンにコンパイラを備えることで、モジュールを高速に実行する機構を提供する、

これら６点によりモジュールの開発コストを低く抑えた。また

（ｇ）多数のストレージが存在するときに、モジュールを配布するために管理コンピュータを用いることにより、ＡＮＳを多数利用する際の総所有コストを低く抑えた。

【図面の簡単な説明】

【図１】

本実施例の全体構成を示すブロック図。

【図２】

サーバ接続型ストレージの構成図。

【図３】

ネットワーク接続型ストレージの構成図。

【図４】

高機能サーバ接続型ストレージの構成図。

【図５】

高機能ネットワーク接続型ストレージの構成図。

【図６】

アクティブネットワークストレージの内部構成図。

【図７】

モジュールの内部構成図。

【図８】

オブジェクト記述データを用いるオブジェクトアクセスモジュールの内部構成図。

【図９】

プロテクションモジュールの内部構成図。

【図１０】

オブジェクト記述データ（表形式）の構成図。

【図１１】

オブジェクト記述データ（表形式）の使用例を示す図。

【図１２】

オブジェクト記述データ（パーザ形式）の構成図。

【図１３】

オブジェクト記述データ（パーザ形式）の使用例を示す図。

【図１４】

オブジェクト記述データ（パターンマッチ形式）の構成図。

【図１５】

保護ポリシーの構成図。

【図１６】

表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート。

【図１７】

パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート（１）。

【図１８】

パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート（２）。

【図１９】

パーザ内で用いられるｆｏｕｎｄＯｂｊｅｃｔ処理手順のフローチャート。

【図２０】

ＡＮＳ利用コンピュータからＡＮＳへのモジュール登録動作の構成図。

【図２１】

ＡＮＳ利用コンピュータからＡＮＳへの、ＡＮＳ管理コンピュータを経由したモジュール登録動作の構成図。

【図２２】

ＡＮＳ管理表の構成図。

【図２３】

ＡＮＳによるストライプ入出力の構成図

【符号の説明】

１０１：全体

１０２、１０２’、…：コンピュータ、１０３：ネットワーク、

１０４：アクティブネットワークストレージ（ＡＮＳ）、

１０５：ＡＮＳコントローラ（ＡＮＳＣ）、

１０６：ネットワーク制御部、１０７：モジュール実行部、

１０８：バッファ管理部、１０９：バッファメモリ、

１１０：ディスク制御部、１１１：ディスク、

１１２：１１２’、１１２”、…：モジュール、

１１３：１１３：機能モジュール、

１１４：１１４：オブジェクトアクセスモジュール、

１１５：ブロックアクセスモジュール、

１１６：プロテクションモジュール、１１７：ロックモジュール、

１１８：ＡＮＳインタフェース、１１９：モジュール登録・削除

１２０：メソッド起動

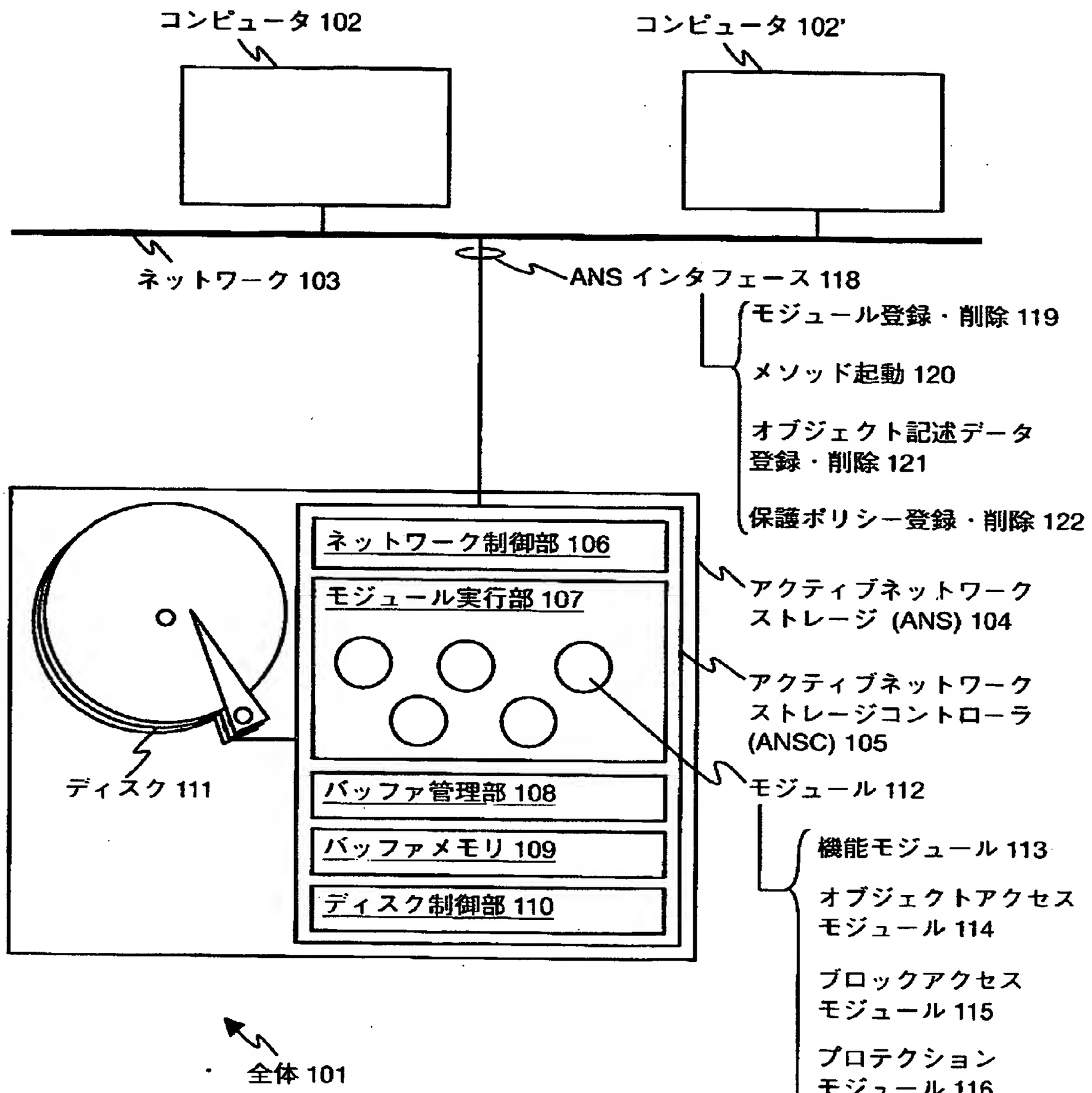
１２１：オブジェクト記述データ登録・削除

１２２：保護ポリシー登録・削除。

【書類名】　図面

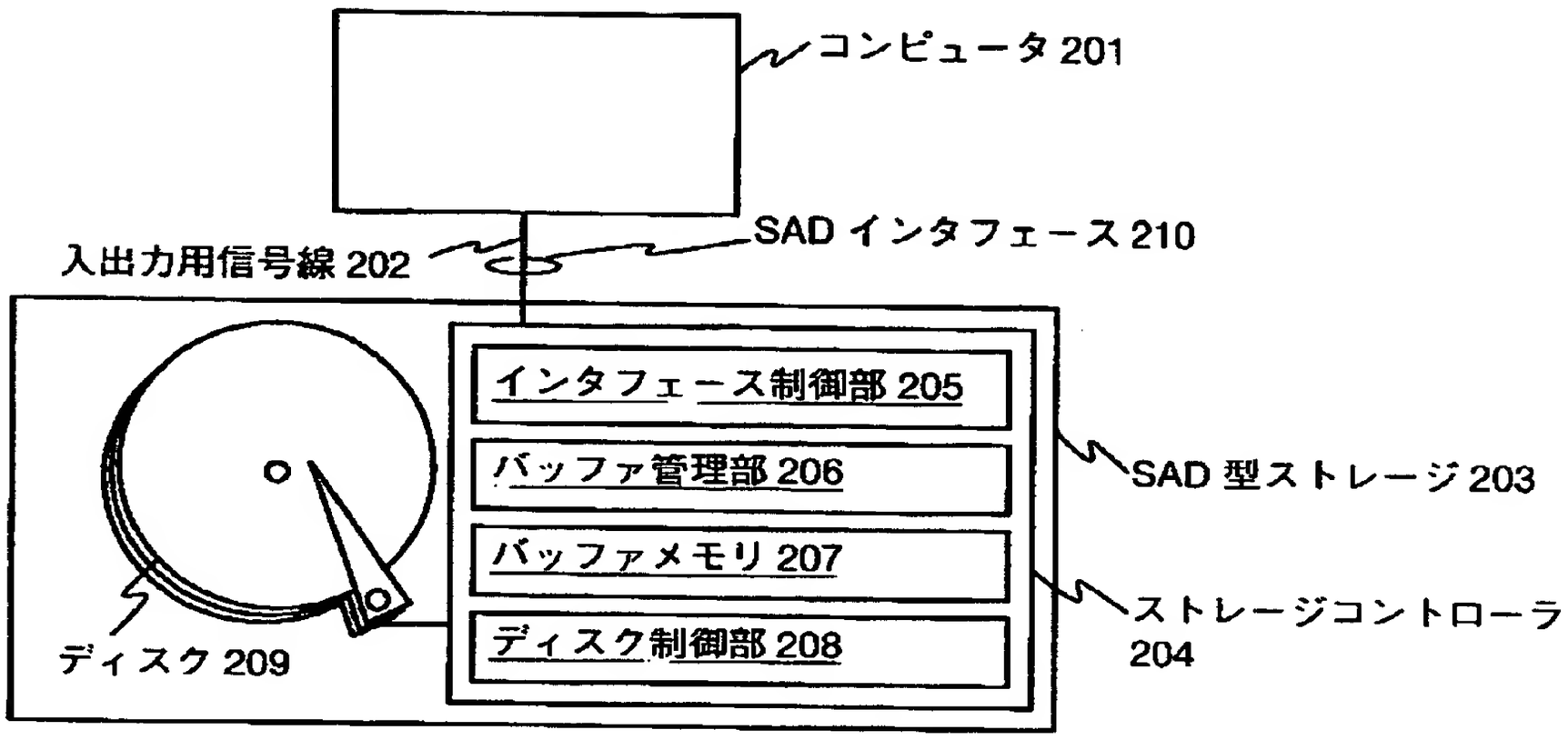
【図１】

図 1


コンピュータ 102
コンピュータ 102'
ネットワーク 103
ANS インタフェース 118
モジュール登録・削除 119
メソッド起動 120
オブジェクト記述データ
登録・削除 121
保護ポリシー登録・削除 122
ネットワーク制御部 106
モジュール実行部 107
アクティブネットワーク
ストレージ (ANS) 104
アクティブネットワーク
ストレージコントローラ
(ANSC) 105
モジュール 112
ディスク 111
バッファ管理部 108
バッファメモリ 109
ディスク制御部 110
機能モジュール 113
オブジェクトアクセス
モジュール 114
ブロックアクセス
モジュール 115
プロテクション
モジュール 116
ロックモジュール 117
全体 101

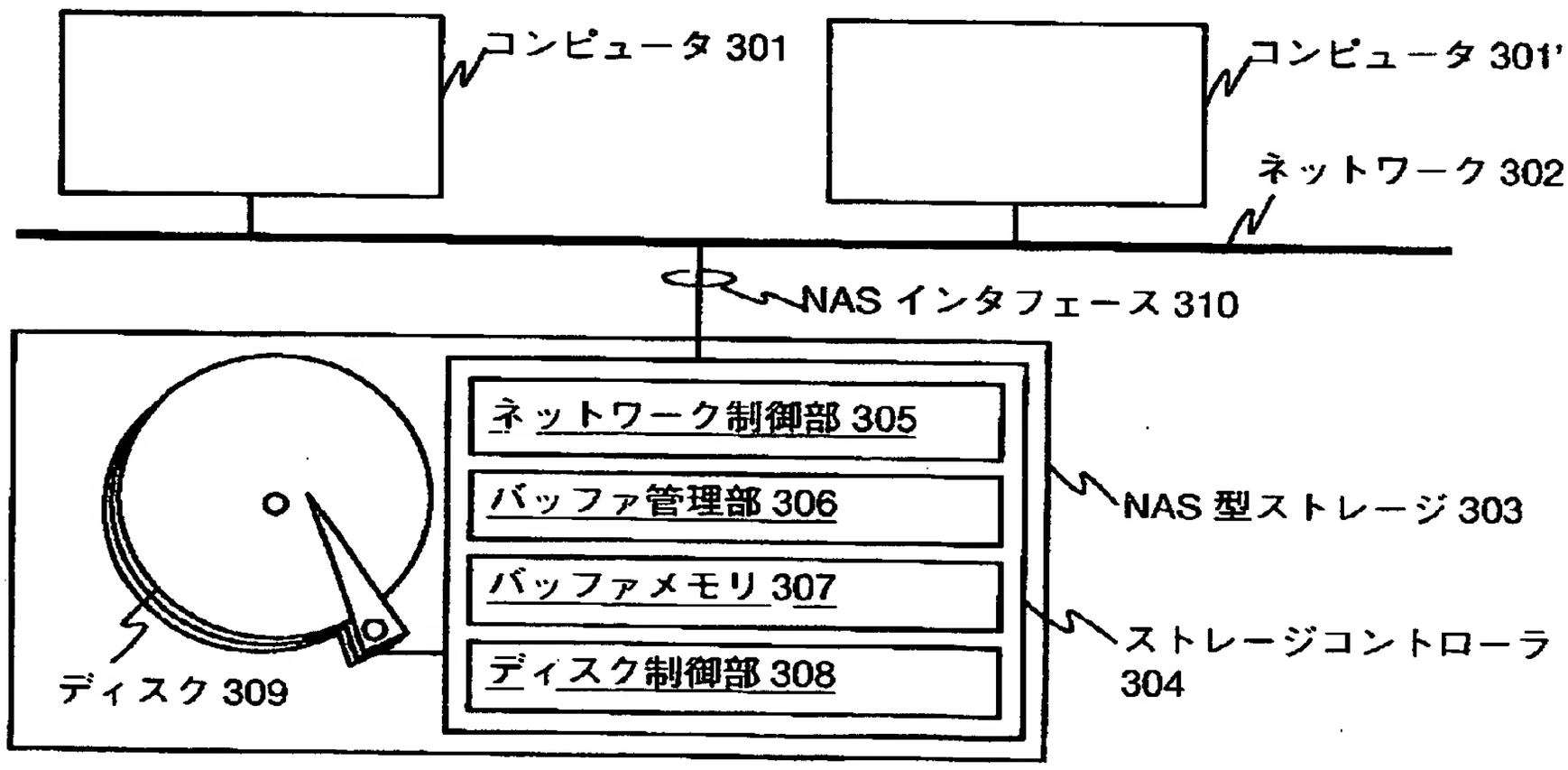
【図２】

図２


コンピュータ 201
SAD インタフェース 210
入出力用信号線 202
インタフェース制御部 205
バッファ管理部 206
バッファメモリ 207
ディスク制御部 208
ディスク 209
SAD 型ストレージ 203
ストレージコントローラ 204

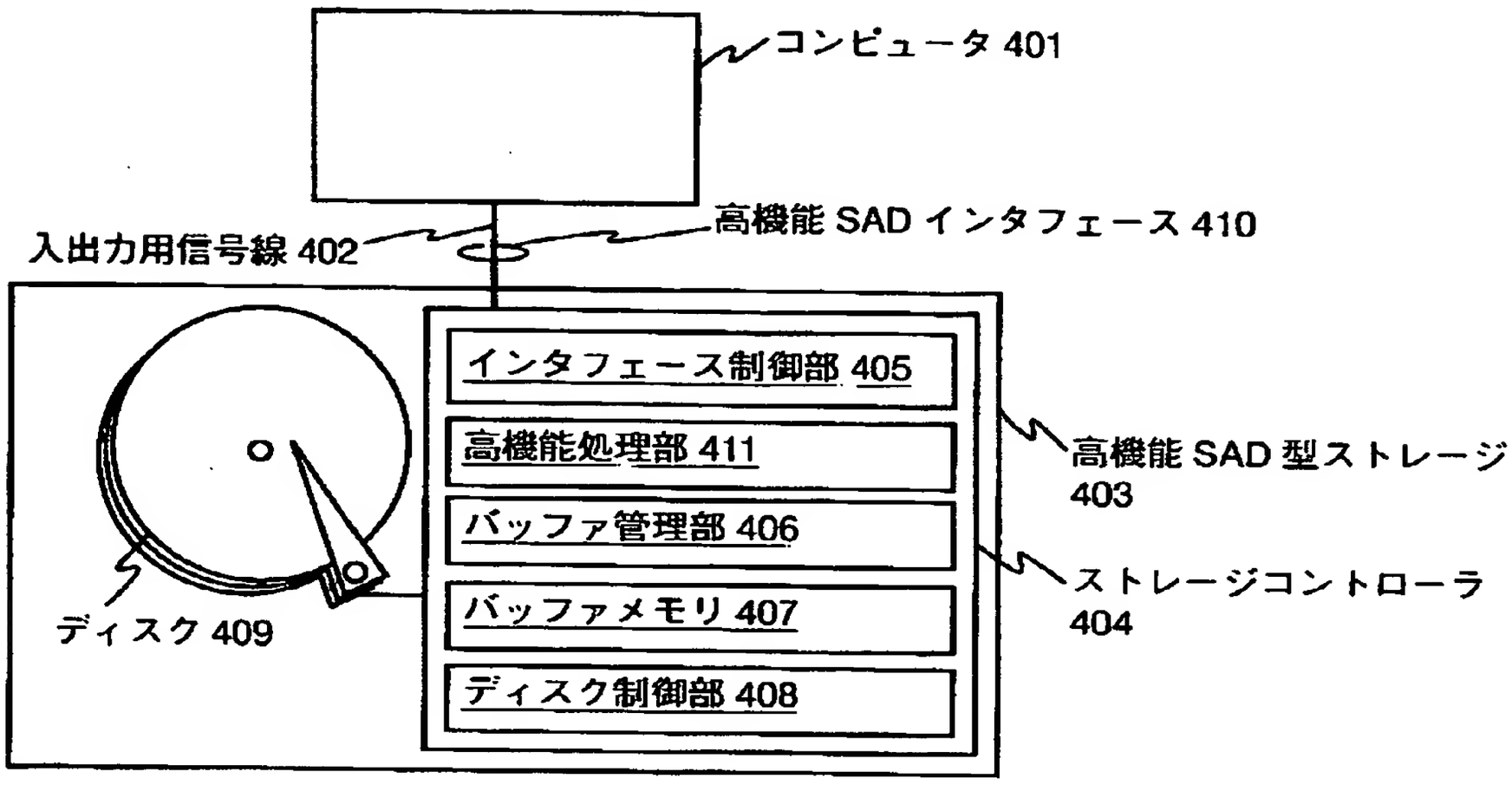
【図３】

図３


コンピュータ 301
コンピュータ 301'
ネットワーク 302
NAS インタフェース 310
ネットワーク制御部 305
バッファ管理部 306
バッファメモリ 307
ディスク制御部 308
ディスク 309
NAS 型ストレージ 303
ストレージコントローラ 304

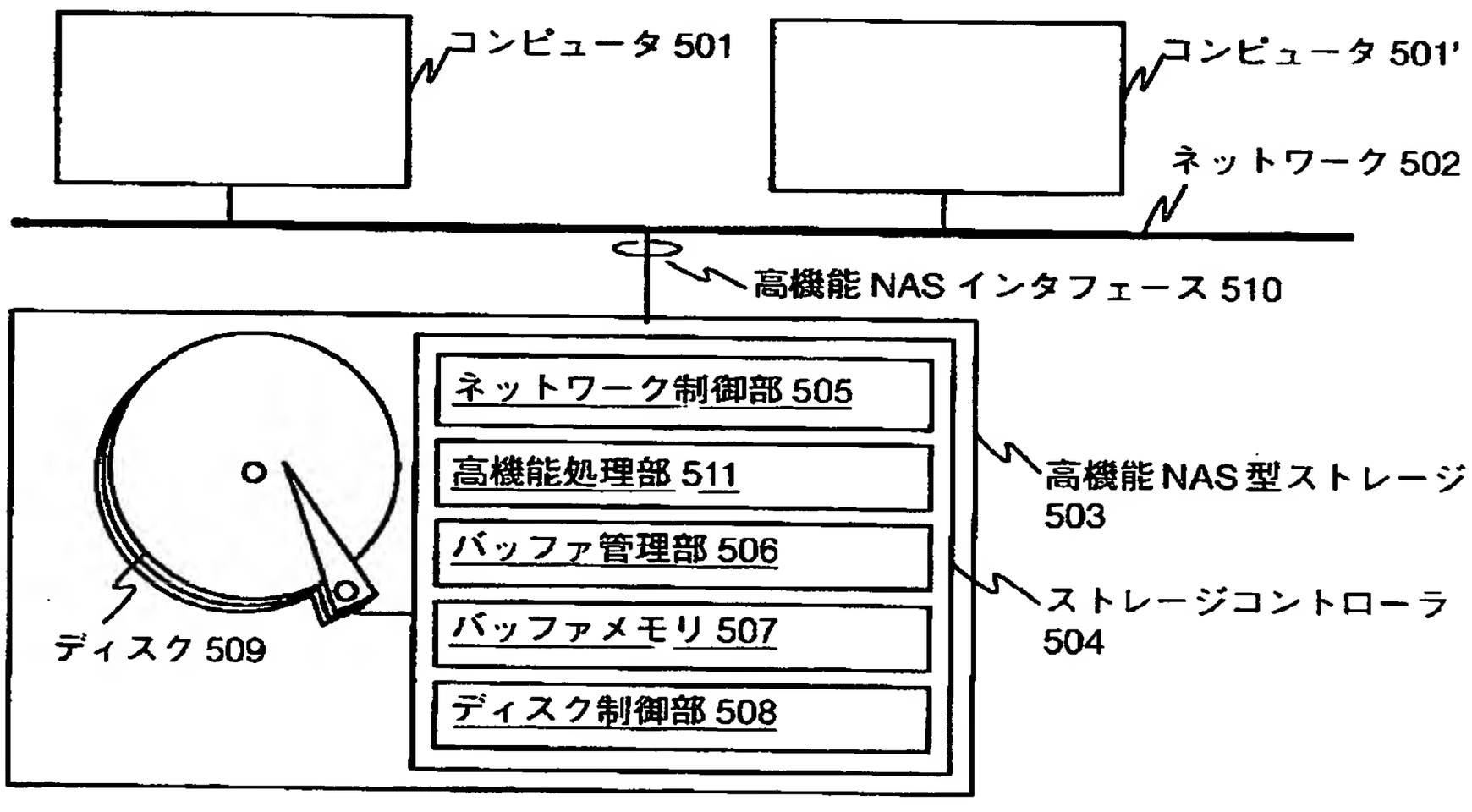
【図４】

図４


コンピュータ 401
高機能 SAD インタフェース 410
入出力用信号線 402
インタフェース制御部 405
高機能処理部 411
バッファ管理部 406
バッファメモリ 407
ディスク制御部 408
ディスク 409
高機能 SAD 型ストレージ 403
ストレージコントローラ 404

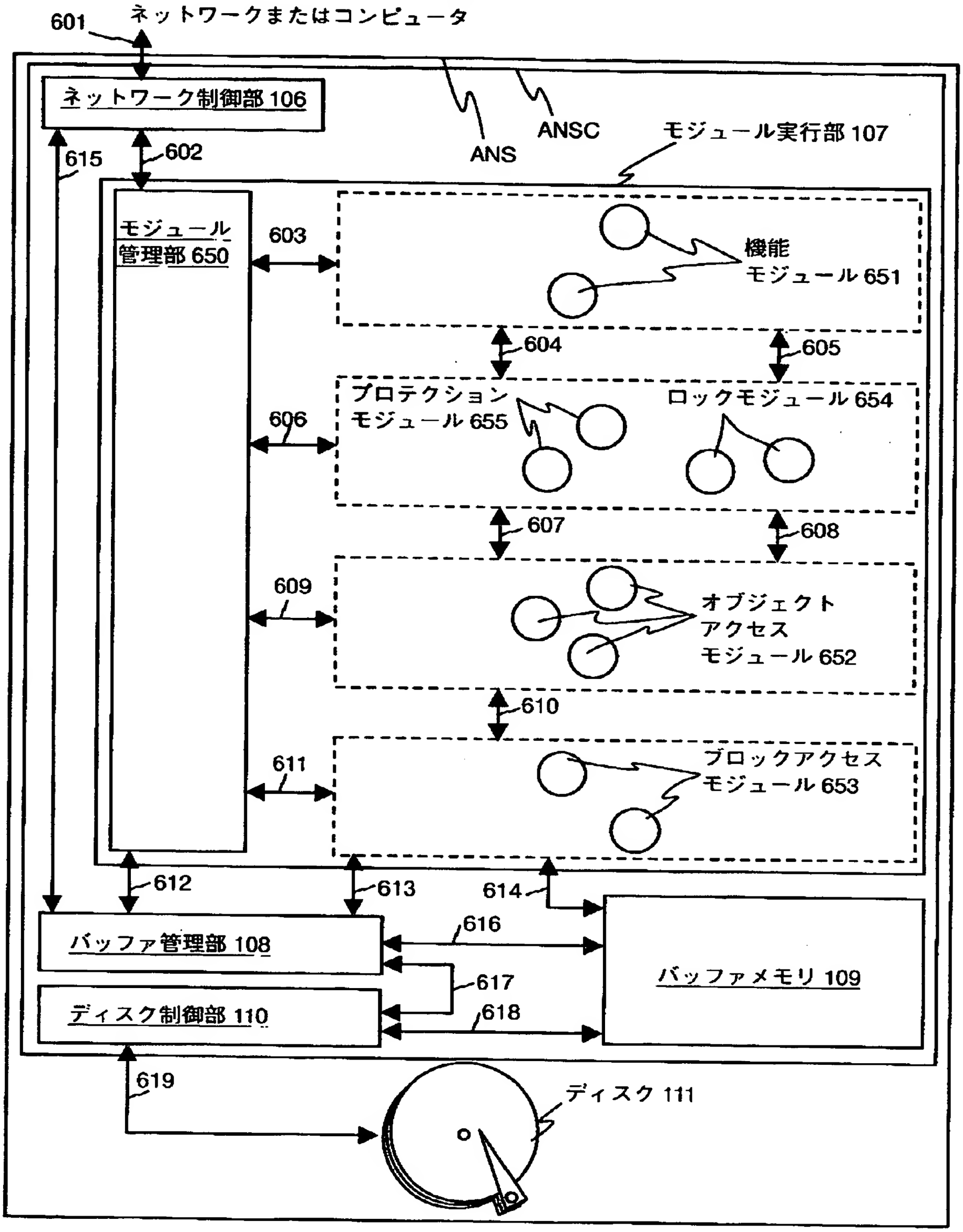
【図５】

図５


コンピュータ 501
コンピュータ 501'
ネットワーク 502
高機能 NAS インタフェース 510
ネットワーク制御部 505
高機能処理部 511
バッファ管理部 506
バッファメモリ 507
ディスク制御部 508
ディスク 509
高機能 NAS 型ストレージ 503
ストレージコントローラ 504

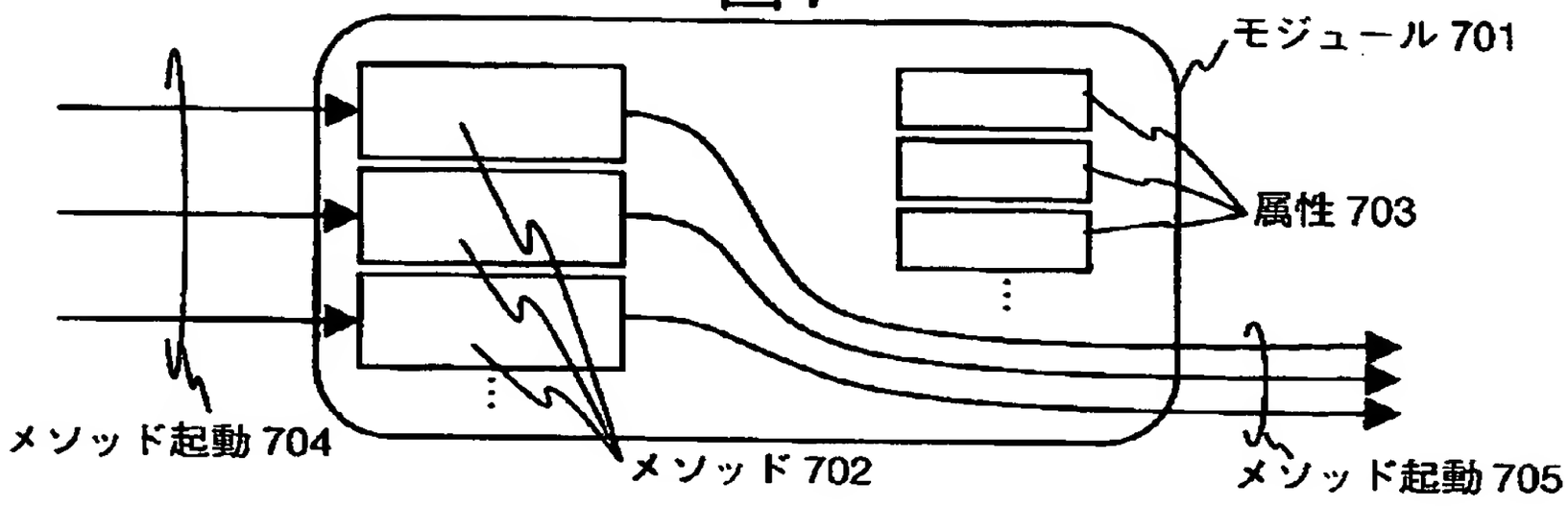
【図６】

図６


601
ネットワークまたはコンピュータ
ネットワーク制御部 106
615
602
ANS
ANSC
モジュール実行部 107
モジュール
管理部 650
603
機能
モジュール 651
604
605
プロテクション
モジュール 655
ロックモジュール 654
606
607
608
609
オブジェクト
アクセス
モジュール 652
610
611
ブロックアクセス
モジュール 653
612
613
614
616
バッファ管理部 108
617
ディスク制御部 110
618
バッファメモリ 109
619
ディスク 111

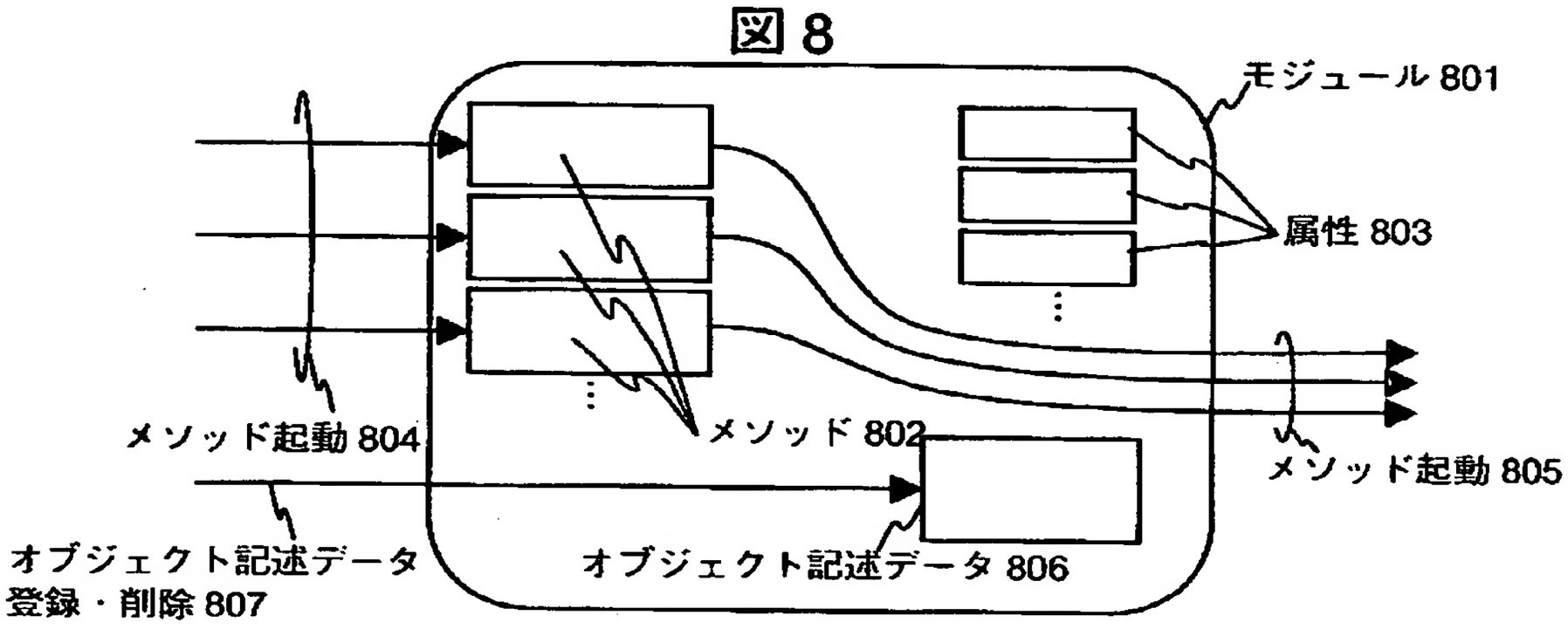
【図７】

図７



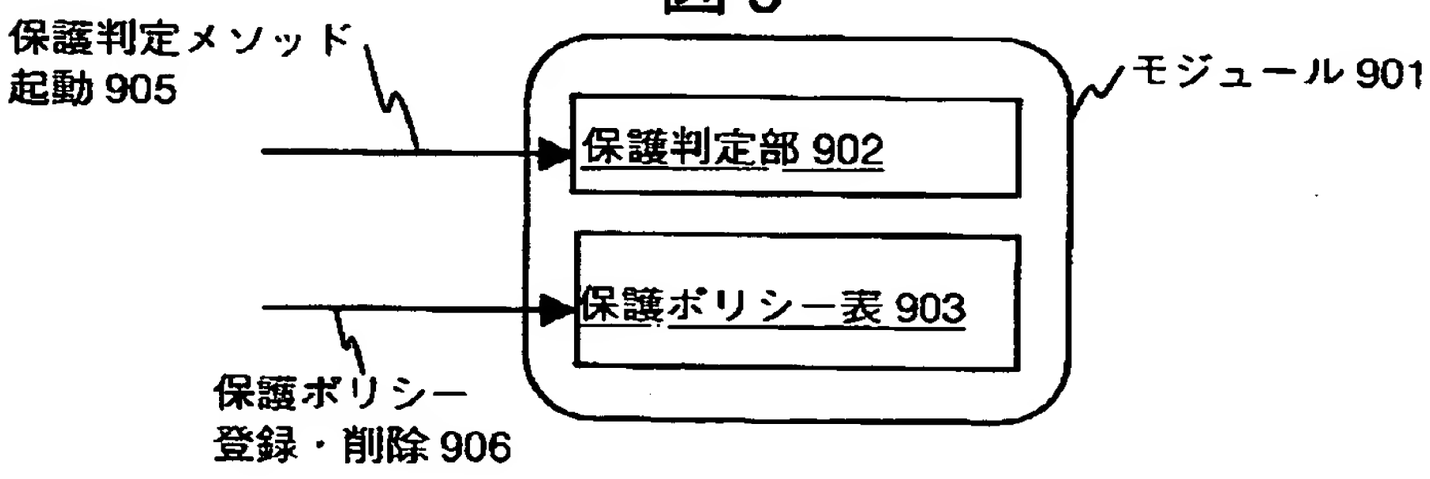
【図８】

図８



【図９】

図９



【図１０】

図 10

オブジェクト記述データ (表形式) 1001

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
		⋮			

【図１１】

図 11

UFS_inode オブジェクト 1101

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
di_mode	short	0	2	1	null
di_nlink	short	2	2	1	null
di_uid	short	4	2	1	null
di_gid	short	6	2	1	null
di_size	long	8	4	1	null
di_addr1	BLOCK	12	3	10	UFS_data
di_addr2	BLOCK	42	3	1	UFS_indirect1
di_addr3	BLOCK	45	3	1	UFS_indirect2
di_addr4	BLOCK	48	3	1	UFS_indirect3
di_gen	byte	51	1	1	null
di_atime	long	52	4	1	null
di_mtime	long	56	4	1	null
di_ctime	long	60	4	1	null

UFS_data オブジェクト 1102

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
data	byte	0	1	4096	null

UFS_indirect1 オブジェクト 1103

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
bid	BLOCK	0	4	1024	UFS_data

UFS_indirect2 オブジェクト 1104

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
bid	BLOCK	0	4	1024	UFS_indirect1

UFS_indirect3 オブジェクト 1105

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007
bid	BLOCK	0	4	1024	UFS_indirect2

【図１２】

図 12

オブジェクト記述データ (パーザ形式) 1201

タグ 1202	初期化コード 1203
コンテクスト 1204	コード 1205
⋮	

【図１３】

図 13

```
create table customer (
        c_id            integer,
        c_name          char(40),
        c_address       varchar(100),
        );
```

1301

132

タグ 1202	初期化コード 1203
コンテクスト 1204	コード 1205
customer()	{ record = 0; column = 0; }
(record())*	null
record()	{ int saved_offset = offset; }
record_size = <long> c_id() c_name() c_address()	{ record++; column = 0; foundObject("record", record_size, saved_off, record); }
c_id()	null
value = <long>	{ column++; foundObject("c_id", value, sizeof(long), record, column); }
c_name()	null
value = <char>[40]	{ column++; foundObject("c_name", value, sizeof(char)*4, record, column); }
c_address()	null
column_size = <long> value = <char>[column_size]	{ column++; foundObject("c_address", value, sizeof(char)*4, record, column); }

【図１４】

図 14

オブジェクト記述データ (パターンマッチ形式) 1401

ファイルフォーマット 1402	タグ 1403	パターン 1404
shell_script	header1	"#!"
	header2	"/bin/sh"
UNIX_executable	header1	ZMAGIC || NMAGIC || OMAGIC
⋮		

【図１５】

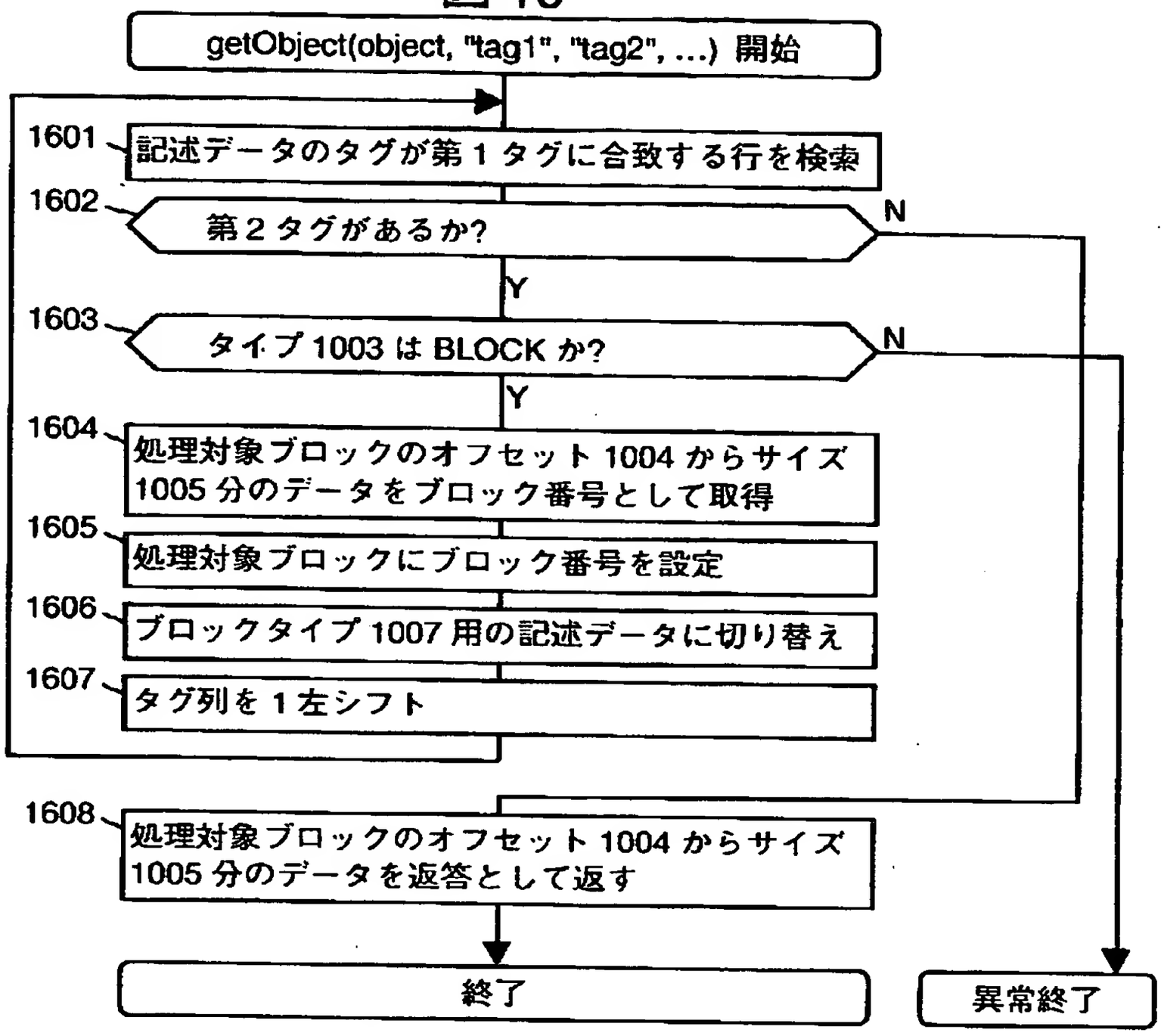
図 15

保護ポリシー記述データ 1501

送信元 1502	送信先 1503	メソッド 1504	許可・禁止 1505
selectionModule	recordAccessModule	getObject("column")	許可
selectionModule	blockAccessModule	getBlock()	禁止
		⋮	

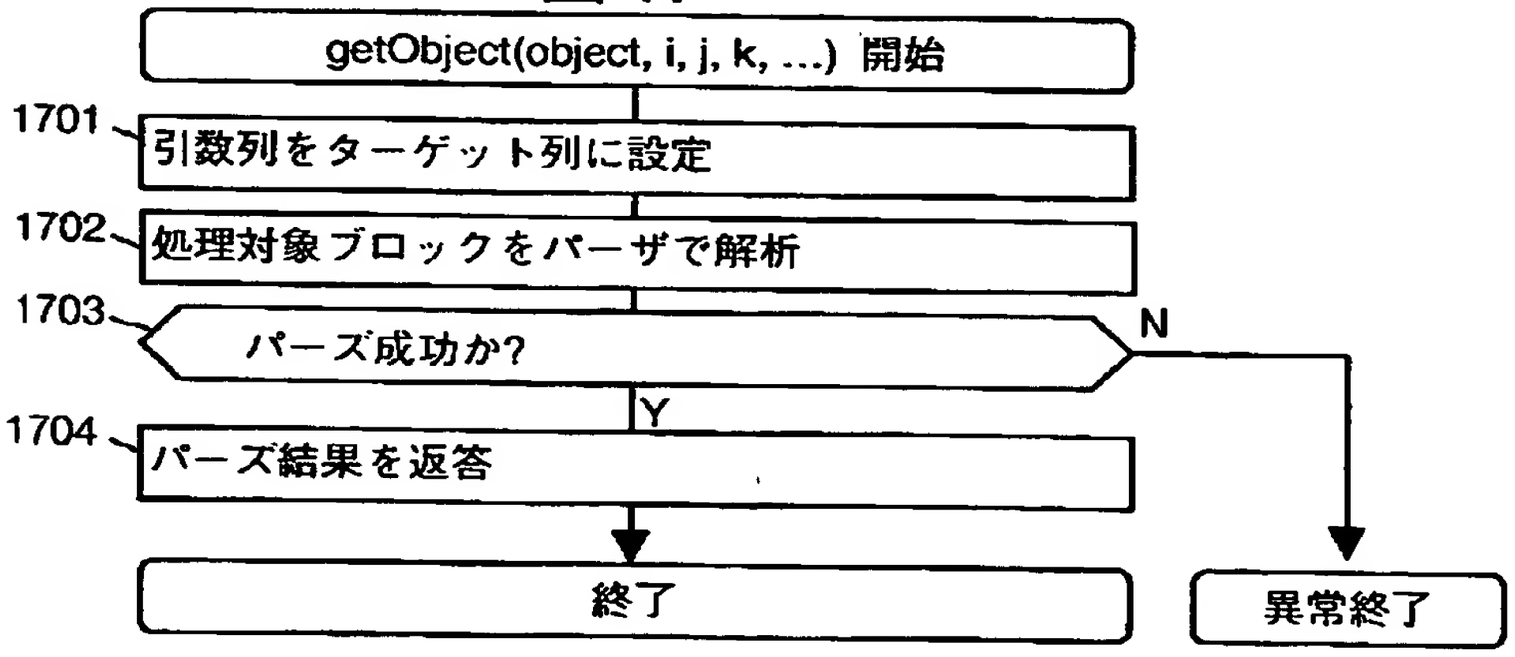
【図１６】

図 16



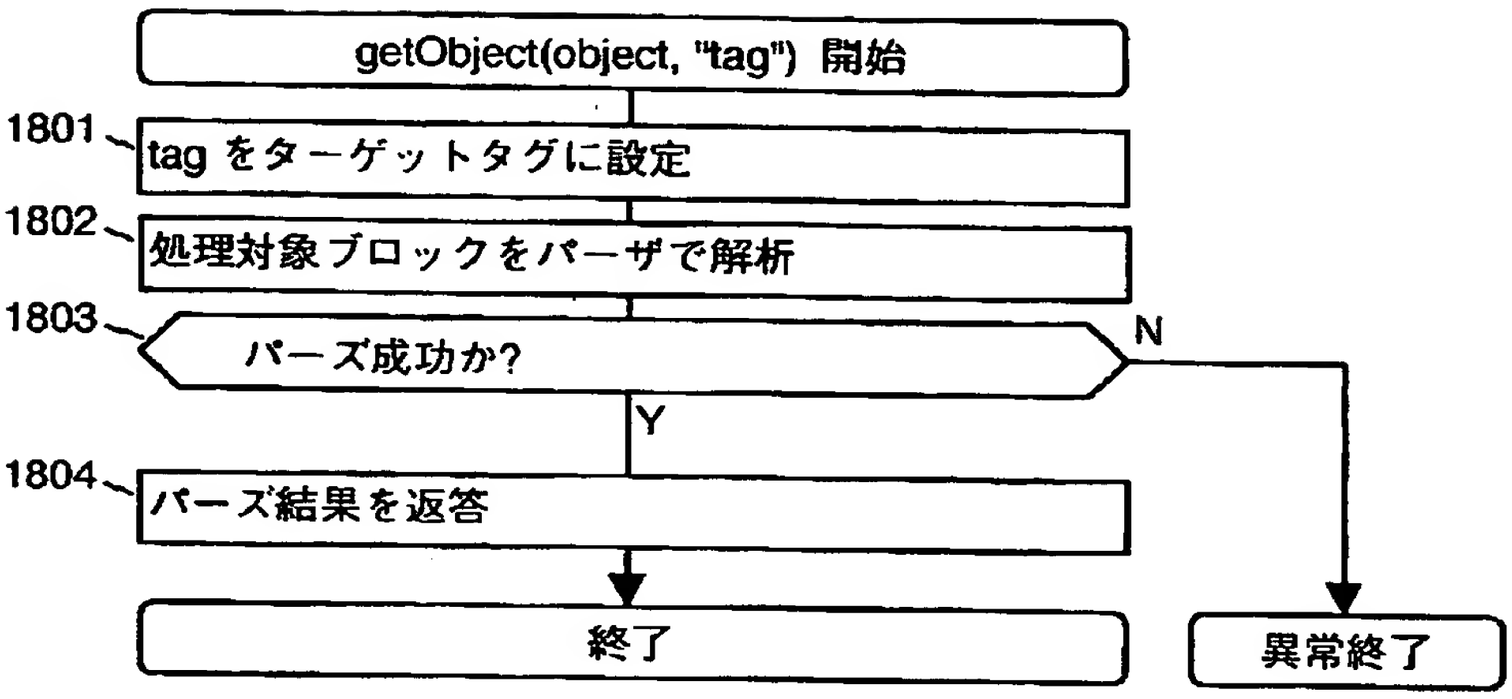
【図１７】

図 17



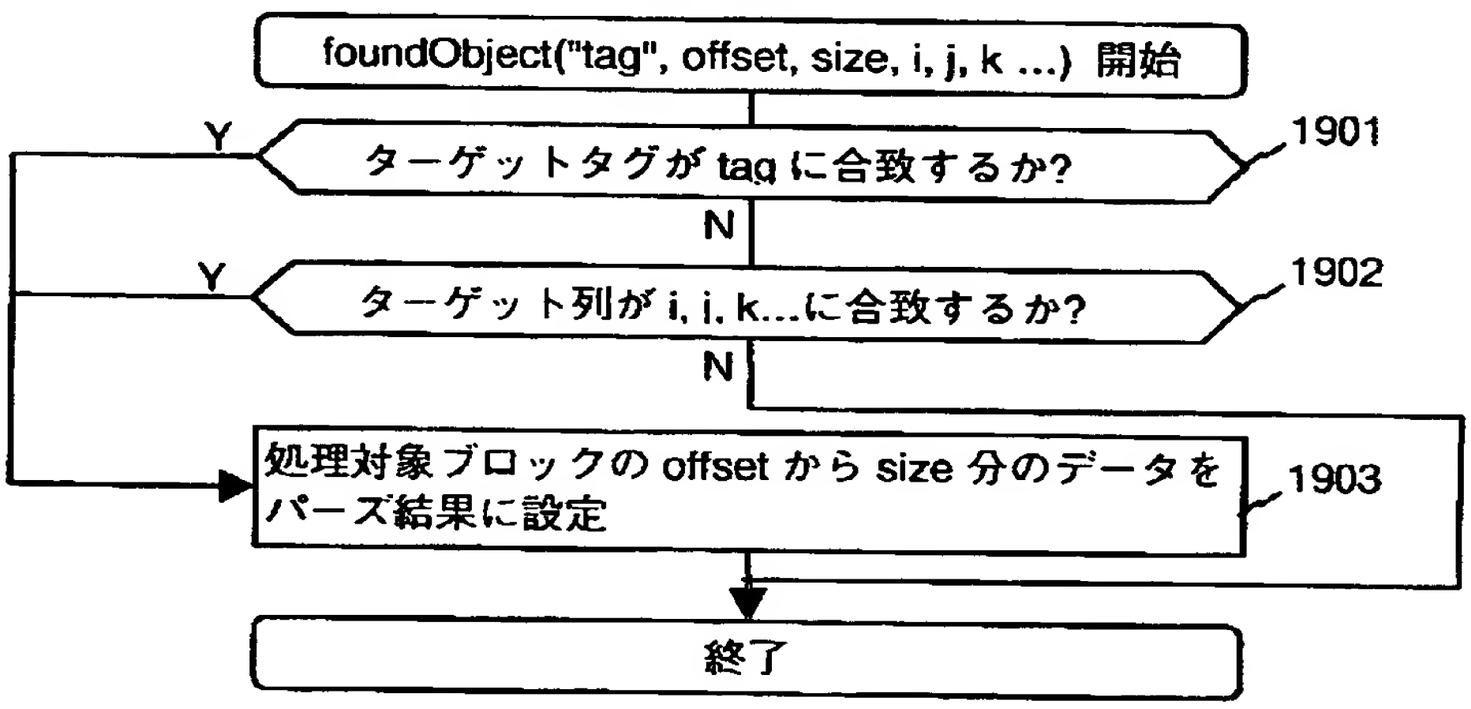
【図１８】

図 18



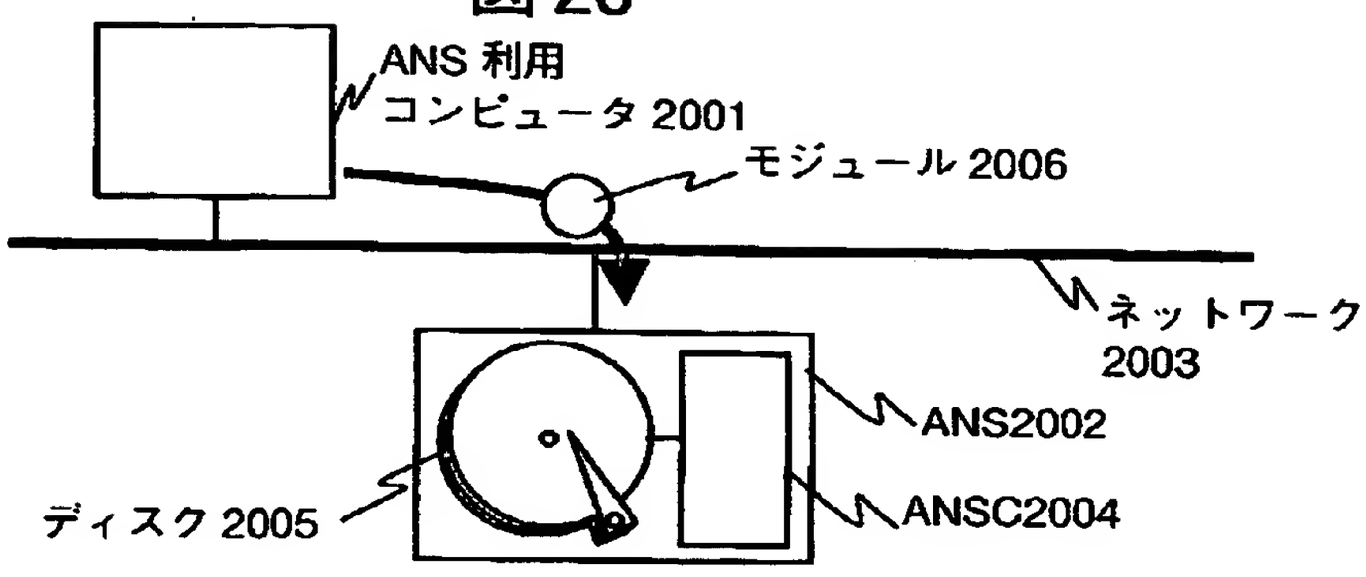
【図１９】

図 19



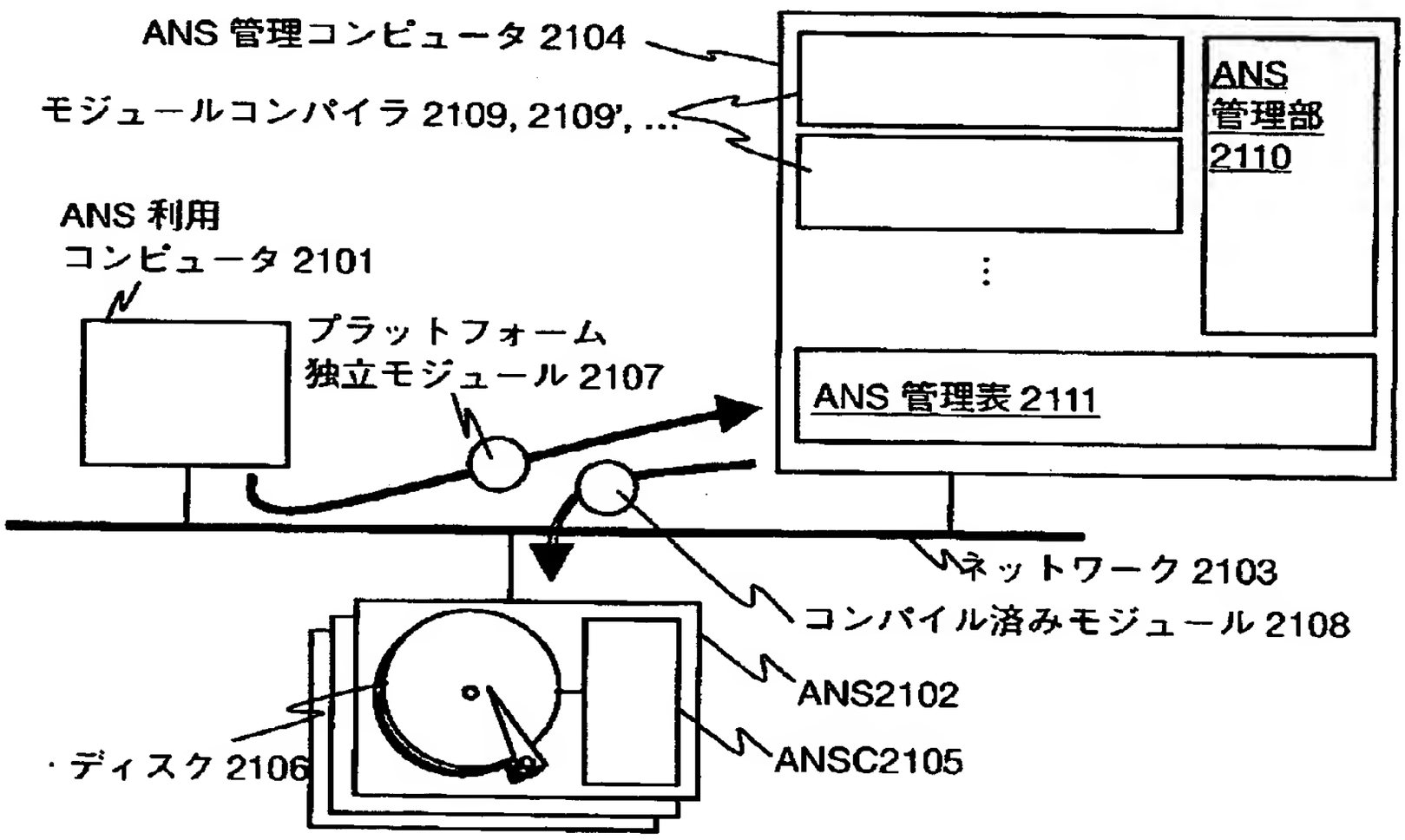
【図２０】

図 20



【図２１】

図 21



【図２２】

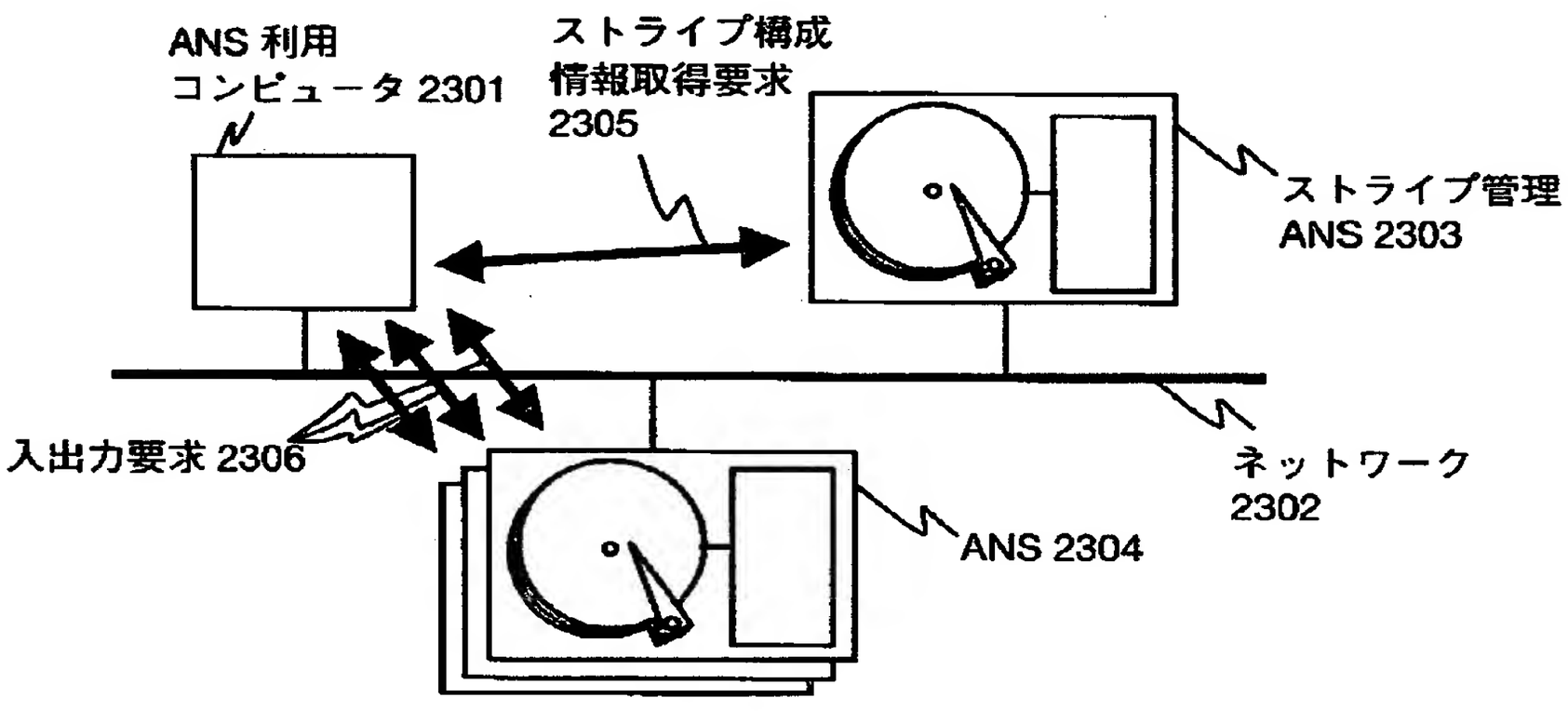
図 22

ANS 管理表 2201

ANS 名 2202	ネットワーク アドレス 2203	機種 2204	コンパイラ 2205
disk1	123.123.11.1	A 社製 342 型	コンパイラ A
disk2	123.123.11.2	B 社製 F01 型	コンパイラ B
		⋮	

【図２３】

図 23



【書類名】　要約書

【要約】

【課題】　広い応用範囲に対して有効な機能を持つストレージを継続的に提供する際の、機能の開発コストと、総所有コストを低く抑える。

【解決手段】　機能を実現するモジュール群の開発コストを低く抑えるため、高機能入出力を実現する際の共通部分となるオブジェクトアクセスモジュール（１１４）をストレージが備え、オブジェクトの二次記憶中への格納方法を宣言的に記述したオブジェクト記述データの登録と削除を行うインタフェース（１２１）をストレージが提供する。総所有コストを低く抑えるため、管理マシン経由でモジュール群をストレージに転送する。

【効果】　広い応用範囲に対して有効な機能を継続的に提供するストレージが、低開発コスト、低総所有コストで提供可能となる。

【選択図】　図１

出証特平１１－３０６８０５４

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　000005108
【住所又は居所】　東京都千代田区神田駿河台四丁目６番地
【氏名又は名称】　株式会社日立製作所
【代理人】　申請人
【識別番号】　100068504
【住所又は居所】　東京都千代田区丸の内１－５－１　株式会社日立製作所　知的所有権本部内
【氏名又は名称】　小川　勝男

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所